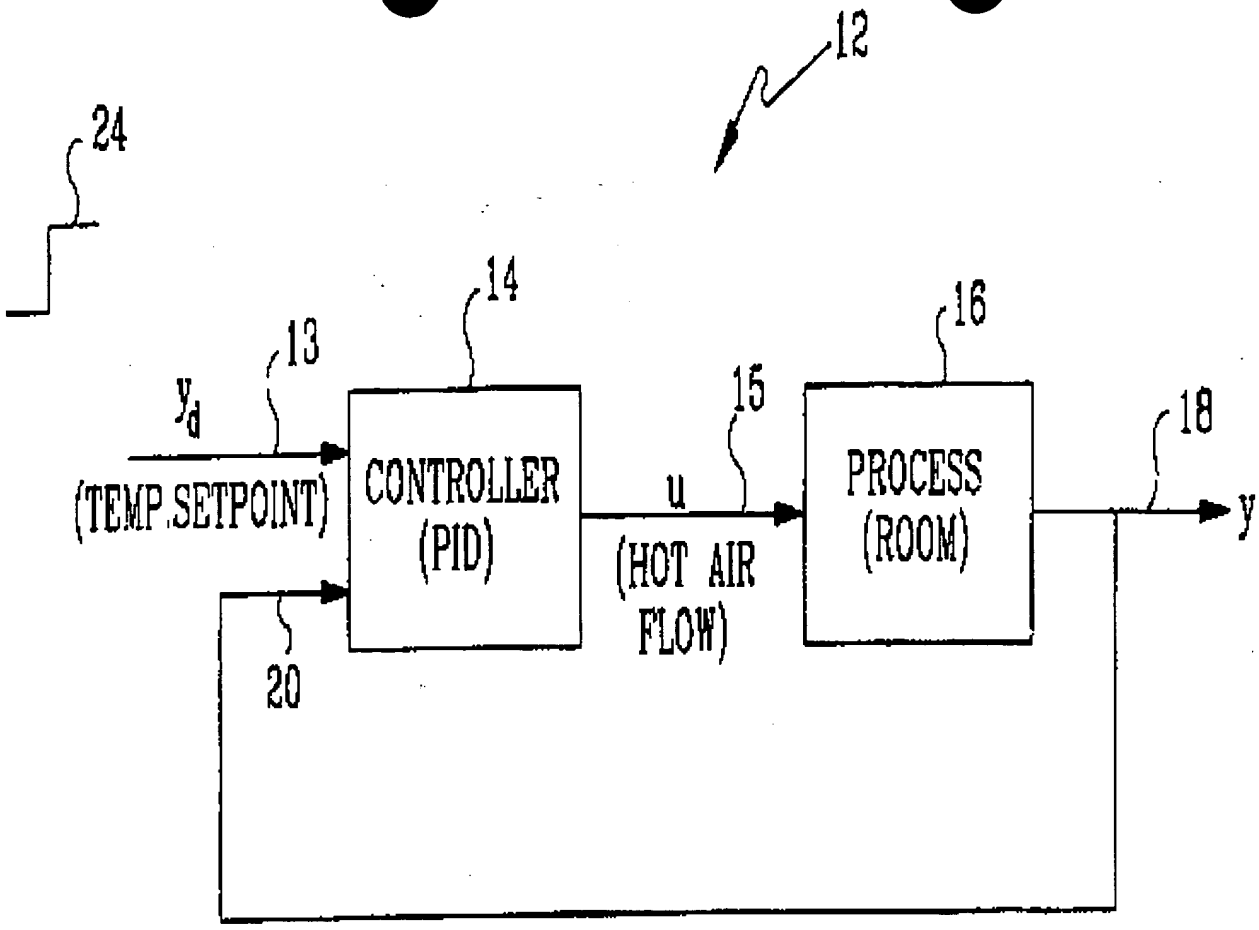
AN: PAT 1998-087132
TI: Automatic turner controller for control system using neural network includes post-processing device for scaling set of normalised tuning parameters into set of scaled tuning parameters at output
PN: WO9800763-A1
PD: 08.01.1998
AB: The device includes a pre-processing device, with an input and an output, for processing a set of parameters at the input into a set of transformed parameters at the output. A non-linear approximator device, has an input connected to the output of the pre-processing device and an output, for operating on the set of transformed parameters to produce a set of normalised tuning parameters at the output. The non-linear approximator device is off-line optimised without advance supervised learning. A post- processing device, has a first input connected to the output of the non-linear approximation device, and a second input and an output, for scaling the set of normalised tuning parameters into a set of scaled tuning parameters at the output. The scaled tuning parameters are fed to the controller for optimally controlling a process.;
PA: (HONE) HONEYWELL INC;
IN: SAMAD T;
FA: WO9800763-A1 08.01.1998; JP2000514217-W 24.10.2000; US5847952-A 08.12.1998; EP907909-A1 14.04.1999; EP907909-B1 03.05.2000; **DE69701878**-E 08.06.2000;
CO: AT; BE; CA; CH; DE; DK; EP; ES; FI; FR; GB; GR; IE; IT; JP; LU; MC; NL; PT; SE; US; WO;
DN: CA; JP;
DR: AT; BE; CH; DE; DK; ES; FI; FR; GB; GR; IE; IT; LU; MC; NL; PT; SE;
IC: G05B-013/02; G05B-013/04; G06F-015/18;
MC: T01-J08A; T01-J16B; T01-J16C1; T06-A05A; T06-A05A1; T06-A06A9; X27-E01A; X27-G02;
DC: T01; T06; X27;
FN: 1998087132.gif
PR: US0671996 28.06.1996;
FP: 08.01.1998
UP: 10.11.2000


12
24
14
16
13
15
18
y_d
(TEMP.SETPOINT)
CONTROLLER
(PID)
u
(HOT AIR
FLOW)
PROCESS
(ROOM)
y
20

02P 04668

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) Übersetzung der europäischen Patentschrift

(97) EP 0 907 909 B1

(10) DE 697 01 878 T 2

(51) Int. Cl.[7]: G 05 B 13/02

(21) Deutsches Aktenzeichen: 697 01 878.4
(86) PCT-Aktenzeichen: PCT/US97/09473
(96) Europäisches Aktenzeichen: 97 929 758.7
(87) PCT-Veröffentlichungs-Nr.: WO 98/00763
(86) PCT-Anmeldetag: 2. 6. 1997
(87) Veröffentlichungstag der PCT-Anmeldung: 8. 1. 1998
(97) Erstveröffentlichung durch das EPA: 14. 4. 1999
(97) Veröffentlichungstag der Patenterteilung beim EPA: 3. 5. 2000
(47) Veröffentlichungstag im Patentblatt: 7. 12. 2000

(30) Unionspriorität:
671996 28. 06. 1996 US

(73) Patentinhaber:
Honeywell, Inc., Minneapolis, Minn., US

(74) Vertreter:
Dipl.-Ing. Dieter Herzbach und Dipl.-Ing. Christoph Sturm, 63067 Offenbach

(72) Erfinder:
SAMAD, Tariq, Minneapolis, US

(84) Benannte Vertragstaaten:
BE, DE, FI, FR, GB, IT

(54) AUF NICHTLINEARER ANNÄHERUNGSMETHODE BASIERENDE AUTOMATISCHE VORRICHTUNG

DE 697 01 878 T 2

AUTOMATISCHER TUNER AUF DER BASIS EINES NICHTLINEAREN APPROXIMATORS

ALLGEMEINER STAND DER TECHNIK

Die vorliegende Erfindung betrifft Tuner für Regler und insbesondere automatische Tuner für Regler. Die Erfindung betrifft ganz besonders Tuner vom Typ des nichtlinearen Approximators. Hintergrundinformationen für die vorliegende Erfindung finden sich in dem am 10. Mai 1994 erteilten US-Patent Nr. 5,311,421 von Masahide Nomura et al. mit dem Titel "Process Control Method and System for Performing Control of a Controlled System by Use of a Neural Network" [Prozeßregelmethode und System zur Durchführung einer Regelung eines geregelten Systems durch Verwendung eines neuronalen Netzes]. Es wird auch auf das deutsche Patent DE 4433332 verwiesen, das die Verwendung eines neuronalen Netzes als Tuner für einen Regler beschreibt.

KURZE DARSTELLUNG DER ERFINDUNG

Die Erfindung ist ein Autotuner zum Abstimmen eines Reglers, wie in den unabhängigen Ansprüchen 1 und 8 definiert. Der Tuner weist einen Vorprozessor zum Umwandeln einer Menge von Eingangssignalen in eine Menge von normierten Parametern auf. Diese Parameter werden in einen nichtlinearen Approximator eingegeben, der offline ohne vorheriges überwachtes Lernen unter Verwendung von mehreren Regelkreissimulationen eingerichtet worden ist. Ein Optimierungsalgorithmus minimiert den Wert einer Kostenfunktion, die von den Regelkreissimulationen abgeleitet ist und die die Güte der Simulationen angibt, um optimierte Parameter des nichtlinearen Approximators zu produzieren. Der nichtlineare Approximator gibt eine Menge von auf der Grundlage des minimierten Werts der Kostenfunktion abgeleiteten normierten Abstimmparametern an einen Nachprozessor aus, der diese Menge von Parametern in Reglerabstimmparameter skaliert, die zu dem Regler gehen. Der nichtlineare Approximator kann auf

neuronalen Netzen oder anderen parametrisierten nichtlinearen Strukturen basieren. Beim Bestimmen von nichtlinearen Approximatorparametern geht es im wesentlichen eigentlich darum, einen Tuner zu entwerfen oder einzurichten. Der Tuner wiederum stimmt den Regler, der einen Regelkreis aufweist, mit entsprechenden Reglerparametern ab.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

Figur 1 zeigt einen grundlegenden Regler und eine Prozeßbeziehung.

Figur 2 zeigt einen Regler mit einem Tuner zusammen mit einem Prozeß.

Figur 3 zeigt einen Prozeß mit einer Sprungfunktionseingabe.

Figur 4 zeigt die Kennlinien einer Steuerkreis-Sprungantwort des Prozesses in Figur 3.

Figur 5 zeigt eine graphische Darstellung mit einer Sprungfunktion mit einem resultierenden Überschwingen und einem schwingenden Ausregeln, die als ein System mindestens zweiter Ordnung mit Verzögerung modelliert werden muß.

Figur 6 offenbart die Regelkreis-Sprungantwortparameter eines Systems.

Figur 7 offenbart einen Tuner mit getrennten Eingängen für Prozeßkennlinienparameter und Steuerkreisgüteparameter.

Figur 8 ist eine graphische Darstellung einer schnellen Ausregelungsgüte mit Überschwingen.

Figur 9 ist eine graphische Darstellung einer langsamen Ausregelungsgüte ohne Überschwingen.

Figuren 10a und 10b zeigen einen automatischen Tuner für Prozesse, die als lineare Systeme erster Ordnung ohne Verzögerung modelliert werden können.

Figur 11 zeigt einen Tuner für ein neuronales Netz, der in Verbindung mit einem parametrisierten Neuroregler verwendet wird.

Figur 12 veranschaulicht einen Tuner nach dem Stand der Technik für ein neuronales Netz mit einem Merkmal eines Algorithmus überwachten Lernens.

Figur 13 ist ein Schemadiagramm eines Rahmens der vorliegenden Erfindung, der nicht auf überwachtem Lernen basiert.

Figur 14a zeigt ein Beispiel für einen Tuner für einen Proportional-Integral-Regler (PI-Regler), der als ein neuronales Netz verkörpert ist, mit Ein-Abstimmknopf-Eingang und -Ausgängen für Proportional- und Integralverstärkungen.

Figur 14b zeigt einen Tuner, der als eine rechnerisch einfache Verkettungsabbildung mit einer Abstimmung verkörpert ist.

Figur 15 zeigt, wie der Tuner von Figur 14 für einen bekannten linearen Prozeß erster Ordnung verwendet werden kann.

Figur 16 ist eine graphische Darstellung für langsames oder schnelles Ausregeln, je nach der Tunereinstellungseingabe.

Figur 17 ist ein Schemadiagramm zum Entwickeln des Tuners von Figur 15 unter Verwendung eines nichtlinearen Optimierungsalgorithmus.

BESCHREIBUNG DER AUSFÜHRUNGSFORM

Figur 1 zeigt einen Proportional-Integral-Differential-Regler (PID-Regler) 12 im einfachen System, der zum Regeln der Temperatur in einem Raum verwendet werden kann. Der Sollwert y_d für die Temperatur wird auf Leitung 13 in den Regler 14 eingegeben, der auf Leitung 15 eine Ausgabe u zum Regeln einer Luftströmung, wie beispielsweise Hitze oder Heißluftströmung, zu Prozeß 16 aufweist. Die Reglerausgabe geht zu dem Ofen über ein Gasventil oder eine andere Ofenregeleinrichtung. Prozeß 16 ist der Raum, dessen Raumtemperatur geregelt wird. Eine Ausgabe y auf Leitung 18 gibt die Temperatur des Raums an. Signal y wird zu einem Eingang 20 des Reglers 14 zurückgeführt. Mit anderen Worten betrachtet sich der

Regler 14 die aktuelle Temperatur auf Leitung 18 und die Solltemperatur (Thermostattemperatur) (y gewünscht) auf Leitung 13 und berechnet mit der Differenz zwischen den Temperaturen die Heizeingabe 15 zu dem Raum bzw. dem Prozeß 16.

"u" kann relativ zu verschiedenen Bereichen in einem einfachen Proportional-Integral-Regler (PI-Regler) gemäß den folgenden Gleichungen gezeigt werden, wobei:

k_c die Proportionalverstärkung für den Regler ist;

k_i die Integralverstärkung für den Regler ist;

t die Zeit ist;

τ eine Integrationsvariable ist;

e(t) der Prozeßausgabefehler zum Zeitpunkt t ist;

u(t) die Reglerausgabe zum Zeitpunkt t ist;

y(t) die Prozeßausgabe zum Zeitpunkt t ist;

$y_d(t)$ die gewünschte Prozeßausgabe bzw. Sollwert des Zeitpunkts t ist;

U(s) die Laplace-Transformierte von u(t) ist;

E(s) die Laplace-Transformierte von e(t) ist;

j ein Summationsindex ist; und

l ein diskreter Zeitindex ist.

$$u(t) = k_c e(t) + k_i \int_{\tau=0}^{\tau=t} e(\tau) d\tau \qquad (1)$$

wobei $e(t) = y_d(t) - y(t)$

$$U(s) = k_c E(s) + \frac{k_i}{s} E(s) \qquad (2)$$

$$u(l) = k_c e(l) + k_i \sum_{j=0}^{l} e(j) \Delta t \qquad (3)$$

Gleichung (1) ist für den kontinuierlichen Zeitbereich angegeben, Gleichung (2) für den Laplace- oder Frequenzbereich und Gleichung (3) für den

diskreten Zeitbereich. Die Proportionalverstärkung (k_c) und Integralverstärkung (k_i) sind auf Werte eingestellt, die für die Anwendung des Regelsystems 14 auf den Prozeß 16 angemessen sind. Δt ist das Abtastintervall.

Die zum Einstellen der Reglerverstärkungen, beispielsweise k_c und k_i, verwendeten Algorithmen werden als "Abstimmalgorithmen" bezeichnet. Der Abstimmalgorithmus wird in einem Tuner 21 von Figur 2 implementiert. Der Tuner gibt die Verstärkungen k_c und k_i über Leitung 22 in den Regler 14 ein. Diese Verstärkungen werden in dem Stand der Technik oftmals durch einen Ansatz des "Trial and Error" eingestellt. Gegenwärtige Tuner in der Technik sind nicht vollständig zufriedenstellend.

Eingaben in den Tuner 21 müssen zwei Arten von Informationen angeben. Die erste Informationsart ist die relevante Dynamik des Prozesses (z.B. Kennlinien des Raums, wie beispielsweise sein Wärmeverlust, Aufheizzeitkonstante und die Verzögerung aufgrund Rohrleitungen vom Ofen zum Raum). Die zweite Informationsart ist die gewünschte Dynamik des Regelkreises (z.B. Zeit bis zum Erreichen des Sollwerts, Ausregelzeit, Überschwingen, Toleranz des Überschwingens und Wichtigkeit des Reduzierens von Regelenergie).

Figur 3 zeigt eine Sprungfunktion 24 von u zum Eingang 15 des Prozesses 16 und Ausgabe y von Leitung 18. Figur 4 zeigt die Kennlinien einer Steuerkreis-Sprungantwort des Prozesses 16. Der Prozeß ist ein Prozeß erster Ordnung mit einem linearen Verzögerungssystem, das üblicherweise für die meisten industriellen und Gebäuderegelprozesse ausreicht. Ein derartiges System weist drei Prozeßmodellparameter K_p, T_p und T_D auf.

Die Verstärkung beträgt $K_p = (y_1 - y_0)/(u_1 - u_0)$.

Die Zeitkonstante $T_p = t_2 - t_1$, (die die Zeit ist, die y benötigt, um 63 Prozent der endgültigen Ausgabe zu erreichen). Die Totzeit oder Verzögerung beträgt

$T_D = t_1 - t_0$. Falls K_p, T_p, T_D geändert werden, dann ändert sich die Kurve für y.

Figur 5 zeigt eine graphische Darstellung mit einer Sprungfunktion, die ein resultierendes Überschwingen und ein schwingendes Ausregeln 25 aufweist, die als ein System mindestens zweiter Ordnung mit Verzögerung relativ zu dem Eingang 24 modelliert werden muß.

Figur 1 zeigt ein Regelkreissystem 12 mit einem Sprungfunktionssignal 24 für Eingang 13. Figur 6 offenbart die Regelkreis-Sprungantwortparameter von System 12.

Die Prozeßüberschwingweite beträgt $y_{os} = (y_2 - y_1)/(y_1 - y_0)$

Die Ausregelzeit beträgt $t_{ST} = t_1 - t_0$

Die Anregelzeit beträgt $t_{RT} = t_2 - t_0$

Die Regelenergie beträgt $$E_c = \sum_{\tau=t_0} \left(u(\tau) - u(\tau-1)\right)^2$$

Ein wichtiges Ziel beim Abstimmen eines Reglers besteht darin, diese und andere Regelkreis-Antwortparameter zu manipulieren. So ist beispielsweise großes Überschwingen in der Regel unerwünscht, aber oftmals unvermeidlich, falls schnelle Ausregelzeiten erforderlich sind (die Ausregelzeit ist die Zeit nach der Sollwertänderung, nach der die Prozeßausgabe y innerhalb einer bestimmten Größe, z.B. 95% der gewünschten Ausgabe oder des Sollwerts y_d liegt, wobei der Prozentsatz relativ zu der Größe der Sollwertänderung berechnet wird). Bei schnellem Ausregeln sind die Regelenergien, was ebenfalls typisch ist, hoch. Die Antwortparameter weisen zueinander und zu den Abstimmparametern eine komplizierte Beziehung auf, und es liegen keine Formeln vor, die diese Beziehung präzise kennzeichnen können.

Verschiedene Abstimmethoden nehmen verschiedene Arten von Tunereingaben der relevanten Dynamik des Prozesses und gewünschten Dynamik des Kreises und stellen Verstärkungen für verschiedene Arten von Reglern bereit. Die vorliegende Erfindung des

Abstimmens kann für jede Art von linearem oder nichtlinearem Regler verwendet werden, nicht nur für einen PID-Typ. Tunereingaben können wie gewünscht zugeschnitten werden, und der Tuner kann in jedem beliebigen gewünschten Ausmaß robust gemacht werden. Die Robustheit ist die gnädige Toleranz gegenüber der Unbestimmtheit des Wissens über das System. Sie ist auch die gnädige Toleranz gegenüber einer Änderung oder Drift im System.

Zwei Arten von Parametern sind "nominal" und "geschätzt". So ist beispielsweise K_p die nominale oder tatsächliche Verstärkung und $\hat{K}_p$ ist die geschätzte Verstärkung. Regler sind ausgelegt, die Tatsache zu berücksichtigen, daß $K_p \neq \hat{K}_p$ ist.

Merkmale der vorliegenden Erfindung sind die Verwendung von einem oder mehreren nichtlinearen Approximatoren in dem Tuner und die Verwendung eines allgemeinen Optimierungssystems. Die folgenden Arten von nichtlinearen Approximatoren werden explizit berücksichtigt:

- Herkömmliche neuronale Multilayer-Perzeptron-Netze
- Rechnerisch einfache, nichtlineare sigmoidale Verkettungsabbildungen
- Radial-Basisfunktionsnetze
- Funktionsverbindungsnetze
- CMAC-Netze (cerebellar model articulation controller networks)
- Fuzzy-Logik-Modelle
- Wavelet-Netze
- Polynomausdrücke
- Spezifische nichtlineare parametrisierte Strukturen auf der Basis menschlicher Eingebung über Reglerabstimmung

Diese und andere relevante Approximatorstrukturen haben den Gesichtspunkt gemeinsam, daß sie nichtlineare Abbildungen implementieren, deren Form durch Manipulieren von mit dem Approximator verknüpften Parametern nachgestellt werden kann. Somit kann allgemein ein nichtlinearer Approximator, wie er in der

vorliegenden Erfindung verwendet wird, wie folgt ausgedrückt werden:

$$p_c = F(I;p_a)$$

wobei p_c der Vektor von Reglerabstimmparametern ist, I der Eingabevektor zu dem nichtlinearen Approximator ist und p_a der Parametervektor für den Approximator ist. Beispielsweise kann ein herkömmliches neuronales Multilayer-Perzeptron-Netz die Abbildung F wie folgt implementieren:

$$p_c[j] = \left(1 + \exp\left[-\sum_{i=1}^{i=h} w_{ij}^{0} h_i + b_j^{0}\right]\right)^{-1}$$

wobei die hi dazwischenliegende Systemgrößen ("verborgene Neuronen") sind, berechnet als:

$$h_i = \left(1 + \exp\left[-\sum_{k=1}^{k=n} w_{ki}^{h} I_k + b_i^{h}\right]\right)^{-1}$$

Hier besteht der Parametervektor pa aus den Input-zu-Hidden-Gewichten w_{ki}^{h}, den Hidden-zu-Output-Gewichten w_{ij}^{o} und den Hidden- und Output-Bias-Gewichten b_i^{h} und b_j^{o}. i und k sind Summationsindizes, j ist das Element des Approximatorausgabevektors, h ist die Anzahl verborgener Neuronen und n ist die Anzahl von Elementen des Eingabevektors I. Diese Ausdrücke beziehen sich auf neuronale "Ein-Hidden-Layer"-Netze. Mehrfachschichten aus verborgenen Neuronen können ebenfalls verwendet werden.

Als weiteres Beispiel kann eine rechnerisch einfache nichtlineare sigmoidale Verkettungsabbildung F wie folgt implementieren:

$$p_c[j] = \sum_{i=1}^{i=b} v_{ij} \frac{I.w_i + a_i}{1+|I.w_i + a_i|}$$

wobei der Parametervektor p_a die Vektoren w_i und die Skalare a_i und v_{ij} umfaßt. i ist ein Summationsindex und b ist die Anzahl nichtlinearer Elemente in der Struktur. Der (.) bezeichnet den Punktproduktoperator. Man beachte, daß diese Form des nichtlinearen Approximators, die ansonsten einem neuronalen Netz ähnelt, die Verwendung von transzendenten Funktionen, wie beispielsweise Exponentialfunktionen oder Hyperbeltangensfunktionen, die in der Berechnung aufwendig sind, nicht erfordert. Dieser Approximator ist der Ansatz erster Wahl für die vorliegende Erfindung. Wiederum können mehrfache "Schichten" der Nichtlinearitäten verwendet werden.

Als drittes Beispiel kann ein Radial-Basisfunktionsnetz die Abbildung F wie folgt implementieren:

$$pc[j] = \sum_{i=1}^{i=N} w_{ij} \exp(\|I - \mu_i\| / \sigma_i^2)$$

wobei der Parametervektor p_a die Vektoren μ_i und die Skalare σ_i und w_{ij} umfaßt. i ist ein Summationsindex und N ist die Anzahl Gaußscher Nichtlinearitäten in der Struktur.

Als letztes Beispiel kann ein Polynomausdruck als der nichtlineare Approximator verwendet werden, wie z.B.

$$pc[j] = \sum_{k=1}^{k=n} a_k I_k + \sum_{k=1}^{k=n} \sum_{l=k}^{l=k} b_{kl} I_k I_l$$

wobei der Parametervektor p_a die Skalare a_k und b_{kl} umfaßt und n die Anzahl von Elementen in dem Eingabevektor I ist. k und l sind Summationsindizes.

Figur 7 offenbart ein Reglersystem 30, das einen automatischen Tuner 21, der auf einem nichtlinearen Approximator basiert, mit getrennten Eingängen für Prozeßkennlinienparameter und Regelkreis-

Güteparameter aufweist. Mit letzteren kann die Regelkreisleistung für langsames oder schnelles Ausregeln nachgestellt werden. Die Eingaben 26 und 27 gestatten, daß ein im voraus entworfener Tuner 21 für viele Anwendungen verwendet wird oder für große und zahlreiche Klassen von Anwendungen universell ist. Ein nichtlinearer Approximator 28 wird offline optimiert, aber ohne daß Methoden des überwachten Lernens verwendet werden. Bei dem nichtlinearen Approximator 28 kann es sich um ein neuronales Netz handeln, er wird aber nicht als nichtlineares Regressionsmodell verwendet (wie dies in Nomura et al. der Fall ist -- siehe ihre Fig. 33).

Zusätzlich zu einem neuronalen Netz oder einem anderen nichtlinearen Approximator 28 enthält der automatische Tuner 21 Vorverarbeitungs- und Nachverarbeitungsfunktionen. Zu diesen zählen Skalierfunktionen, die außerhalb des neuronalen Netzes erfolgen. Die Skalierung wird hauptsächlich zu Linearitätszwecken durchgeführt. Prozeßkennlinienparameter 26 (die später als p_p notiert werden) und Regelkreis-Güteparameter 27 (die später als p_j notiert werden) gestatten, daß ein im voraus entworfener automatischer Tuner 21 für eine breite Vielfalt von Anwendungen verwendet wird. Kein Online- oder anwendungsspezifisches Training oder eine derartige Optimierung des neuronalen Netzes oder eine Modifikation anderer Komponenten des automatischen Tuners werden benötigt.

Ein spezifisches Beispiel ist der automatische Tuner 21, der auf einem neuronalen Netz basiert, für Prozesse, die als Prozesse erster Ordnung mit linearen Totzeitprozessen modelliert werden können. Derartige Prozesse können mit drei Prozeßmodellparametern (bzw. Prozeßkennlinienparametern) 26 modelliert werden: die Prozeßverstärkung, die Zeitkonstante und die Totzeit. Ein Regelkreis-Güteparameter wird ebenfalls angenommen: ein Ausregelzeitknopf d_{ST} (dies stellt den Regelkreis-Güteparameter bei 27 in Figur 7 dar), mit dem die

Regelkreisgüte für ein langsames oder schnelles Ausregeln nachgestellt werden kann. Es können andere Regelkreis-Güteparameter am Eingang 27 vorliegen. In diesem Fall kann das neuronale Netz offline ohne die Verwendung von überwachtem Lernen, das im Stand der Technik (wie unten erörtert) verwendet wird, unter Verwendung eines auf Simulation basierenden Optimierungssystems optimiert werden.

Figur 8 zeigt die schnelle Ausregelgüte (d.h. niedriges t_{ST}) mit Überschwingen für einen Eingang 27 mit niedrigem d_{ST}. Figur 9 zeigt die langsame Ausregelgüte (d.h. hohes t_{ST}) ohne Überschwingen für einen Eingang 27 mit hohem d_{ST}. Es kann ein Knopf für d_{ST} vorliegen, der dem automatischen Tuner 21 eine analogähnliche Schwankung liefert, wie beispielsweise von Null bis Eins.

Das Hauptmerkmal der vorliegenden Erfindung ist das Design des nichtlinearen Approximators 28. Während des Betriebs in einer Anordnung ähnlich Figur 7 kann der automatische Tuner 21 eine grundlegende Architektur mit Vorprozessor, nichtlinearem Approximator 28 und Nachprozessor, wie in Figur 10a gezeigt, aufweisen. Man beachte in Figur 10b, daß drei Prozeßkennlinienparameter in den automatischen Tuner 21 eingegeben werden: Abschätzungen der Prozeßverstärkung, der Zeitkonstanten und der Totzeit ($\hat{K}_p$, $\hat{T}_p$ und $\hat{T}_d$ notiert). Die Ausgabe 22 des Tuners 21 besteht aus Reglerparametern (bei einem PID-Regler beispielsweise würden diese die Proportional-, Integral- und Differentialverstärkungen sein) p_c. In Figur 10b ist p_c^{bias} eine interne Systemgröße für den automatischen Tuner 21, die aus Grundlinien-Reglerparametern besteht, die dann durch das neuronale Netz oder einen nichtlinearen Approximator 28 auf kontextsensitive Weise modifiziert werden. Man beachte, daß durch intelligentes Strukturieren des Autotuners 21 die Anzahl der Eingaben, die für den nichtlinearen Approximator 28 benötigt werden, in diesem Fall auf Zwei reduziert wird. $\hat{T}_d/\hat{T}_p$ auf Leitung 46 und d_{ST} auf Leitung 29 werden in den nichtlinearen Approximator 28

eingegeben. Eine Ausgabe 31, $\tilde{p}_c$, der ein Regelparameter ist, wird einem Summenpunkt 32 zugeführt. Ein Bias p_c wird auf Leitung 33 zu $\tilde{p}_c$ beim Summierer 32 hinzuaddiert, der auf Leitung 34 einen Wert $p_c$nom ausgibt, der der Vektor von nominalen (unskalierten) Regelparametern ist und der in den Skaliermechanismus 35 eingegeben wird. Die Ausgabe 22 der Skalierung 35 geht zu dem Regler 14.

Der Tuner 21 kann in einer Regelstruktur mit sowohl linearen als auch nichtlinearen Komponenten verwendet werden. So kann der Tuner 21 beispielsweise in Verbindung mit einem parametrisierten Neuroregler 36 verwendet werden, wie in Figur 11 gezeigt.

Der auf einem neuronalen Netz basierende automatische Tuner 21 wird offline unter Verwendung eines Designrahmens optimiert. Es ist ein "evolutionärer Rechen"-Algorithmus entwickelt worden, der Aspekte von genetischen Algorithmen integriert. Auch kann eine Gradientensuche aktiviert werden, wobei Gradienten numerisch berechnet werden.

Der flexible Charakter des Designrahmens gestattet eine Optimierung im Hinblick auf Kriterien, die herkömmlichere Ansätze selten gestatten. Insbesondere brauchen Gütekriterien nicht quadratisch zu sein, noch nicht einmal differenzierbar; Regelstrukturen können willkürlich nichtlinear sein; und Robustheit kann explizit berücksichtigt werden. Der gegenwärtige Ansatz für die Optimierung des auf einem neuronalen Netz basierenden automatischen Tuners 21 erfordert kein "vorheriges Lernen", bei dem Eingabe-Ausgabe-Kombinationen zuerst unter Verwendung eines Separatoroptimierungsprogramms zusammengestellt werden müssen, wobei das neuronale Netz 28 dann unter Verwendung von überwachtem Lernen trainiert wird. Der gegenwärtige Ansatz kann somit dem Ansatz von Nomura et al. (US-Patent 5,311,421) gegenübergestellt werden, in dem die Verwendung eines neuronalen Netzes zum Abstimmen ebenfalls offenbart ist, in dem aber die Entwicklung des neuronalen Netzes einen Algorithmus

überwachten Lernens erfordert, für den entsprechende Lehrdaten gesammelt werden müssen.

Man kann das überwachte Lernen von Nomura et al. für den neuronalen Netztuner 37 von Figur 12 verwenden. Für den Tuner 37 können Eingaben mit x und Ausgaben mit P_c bezeichnet werden. d_{ST} ist Teil von x. P_c sind Regelparameter. P_c wird mit der gewünschten Ausgabe P_c* verglichen, und das Ergebnis des Vergleichs beim Vergleicherknoten 38 ist ein Fehler ε, der dem Lernalgorithmus 39 zugeführt wird. Der Algorithmus 39 liefert dann eine Ausgabe zu dem neuronalen Netz 40, um die Gewichtungen für Ausgabenachstellungen nachzustellen. Für dieses überwachte Lernen muß eine große Datenbank von Paaren (x, P_c*) im voraus zusammengestellt werden. P_c* muß für viele unterschiedliche Fälle im voraus berechnet werden. Nomura et al. beinhaltet das Eingeben von Informationen mit den darin enthaltenen Kennlinien in ein neuronales Netz 40 zum "vorherigen Lernen einer Korrelation" zwischen den die Kennlinien enthaltenden Informationen und einem Regelparameter und Bestimmen eines Regelparameters für einen Regler. "Vorheriges Lernen" bedeutet das Bestimmen von optimalen Regelparametern entsprechend verschiedenen Kennlinien des Modells der Kombination Regler-Regelstrecke und nachfolgendes Verwenden der optimalen Regelparameter als lernende Lehrerdaten. In Nomura et al. wird auch angenommen, daß die Eingabe-Ausgabe-Kombinationen als Lerndaten verfügbar sind (Spalte 11, Zeilen 11-17). Die Werte C_j in Gleichung (10) sind ideale Ausgaben, wobei das neuronale Netz darauf trainiert wird, sich an diese anzupassen, und somit müssen die Werte C_j für neuronales Netztraining zur Verfügung stehen. Das neuronale Netz wird als ein nichtlineares Regressionsmodell verwendet, und somit wird das neuronale Netz unter Verwendung der Eingabe-Ausgabe-Trainingsdaten rückwärts angepaßt.

Der gegenwärtige Ansatz basiert nicht auf derartigem überwachtem Lernen. Keine Sammlung von

Paaren (x, P_c^*) wird benötigt, und keine Berechnung von (optimalen) Werten P_c^* ist erforderlich. Das gegenwärtige neuronale Netz wird unter Verwendung von Offline-Optimierung entwickelt, ohne daß es nötig wäre, im Sinne von Nomura et al. "im voraus zu lernen". Es werden keine Eingabe-Ausgabe-Trainingsdaten benötigt.

Es sollte angemerkt werden, daß der Stand der Technik von Nomura et al. dadurch, daß er auf dem Ansatz eines überwachten Lernens basiert, auf bestimmte Arten von neuronalen Netztunern beschränkt ist. Die vorliegende Erfindung vermeidet das überwachte Lernen vollständig zugunsten eines auf Simulation basierenden Optimierungsansatzes, und ein wichtiger, nebenbei auftretender Vorzug dieses Ansatzes besteht darin, daß der Tuner willkürliche, nichtlineare Approximatoren umfassen kann, einschließlich die Art neuronaler Netzstruktur, die in dem angeführten Stand der Technik verwendet wird, aber nicht darauf beschränkt ist.

Das deutsche Patent DE 4433332 beschreibt auch die Verwendung eines neuronalen Netzes als Tuner für einen PID-Regler. Wie bei Nomura et al. müssen zuerst Lerndaten für das neuronale Netz erzeugt werden, bevor das neuronale Netz selbst optimiert werden kann.

Figur 13 zeigt den Designrahmen 41 der vorliegenden Erfindung schematisch. Für jeden Parametervektor p werden Regelkreissimulationen durchgeführt und ein Gütekriteriumswert berechnet. Kein Lernen von Lehrerdaten ist erforderlich. Der Vektor p kann Verstärkungen für eine lineare Regelstruktur enthalten, so daß ein fester Regler für ein bestimmtes Kriterium optimiert werden kann; er kann Parameter für einen auf einem nichtlinearen Approximator basierenden Tuner oder einen neuronalen Netzregler (z.B. parametrisierter Neuroregler (PNC) 36 in Figur 11) enthalten; oder er kann Parameter sowohl für den Tuner als auch den Regler enthalten, so daß beide Module gleichzeitig optimiert werden können. Die Aufgabe der Designaktivität besteht darin, Tuner oder Regler zu entwickeln, die für einen breiten Bereich von

Anwendungen verwendet werden können. Somit müssen für jede Wahl von p zahlreiche Regelkreissimulationen durchgeführt werden. Bei diesen Simulationen werden verschiedene Parameter je nach den Designanforderungen variiert:

- Prozeßmodellparameter p_p, die in das Prozeßmodell (PM) in der Regelkreissimulation eingegeben werden
- Abschätzungsfehler zwischen diesen Prozeßmodellparametern und den Prozeßmodellparameterabschätzungen $\hat{P}_p$, von denen letztere in den Tuner und/oder den Regler eingegeben werden
- Anfangsbedingungen, insbesondere in dem Fall, in dem der Regelkreisregler oder das Prozeßmodell nichtlinear ist
- Sollwertprofile, auch für nichtlineare Regler oder nichtlineare Prozeßmodelle
- Parameter p_J, die die Kostenfunktion beeinflussen - die Verwendung dieser Parameter gestattet die Entwicklung einer Regellösung für eine Klasse von parametrisierten Kriterien

Jede Regelkreissimulation führt zur Berechnung einer elementaren Kostenfunktion J_i. Die Ergebnisse aller für einen Vektor p durchgeführten Simulationen werden zu einer Gesamtkostenfunktion J zusammengesetzt, die auf den Optimierungsalgorithmus zurückgeführt wird:

$$J = f_J(J_1, J_2, \ldots, J_N)$$

wobei N die Anzahl durchgeführter Regelkreissimulationen ist (in der Regel 1000).

Die obigen Absätze beschreiben den allgemeinen Rahmen des Designsystems. Für dieses Projekt sind spezifische Wahlentscheidungen für Optimierungsläufe getroffen worden, nämlich:

- Prozeßmodelle sind erster Ordnung mit Totzeit. Die Prozeßverstärkung K_p ist auf 1,0 normiert, die Prozeßzeitkonstante T_p auf 10,0. Die

Prozeßtotzeit T_d variiert, so daß die Einschränkung in der nächsten Aussage gilt:

- Abschätzungen von K_p, T_p und T_d werden in den Regler oder Tuner des neuronalen Netzes eingegeben. Diese Abschätzungen sind Störungen der eigentlichen Modellparameter, wie unten erörtert. $\hat{T}_d$ ist auf den Bereich $[0,2\hat{T}_d]$ begrenzt.
- Der Parameterschätzwert für einen Parameter p_i ist gleichförmig in $[(1-r)p_i, (1+r)p_i]$ begrenzt, wobei r ein Parameter ist. r ist eine Funktion des Kriteriumsparameters d_{ST}, die zwischen 0 und 1 variieren kann. Diese Funktion ist eine lineare Funktion, so daß der kleinste Wert von r, der erreicht wird, wenn d_{ST} 0 ist, r_{min} ist; und der größte Wert, der d_{ST} gleich 1 entspricht, r_{max} ist. Somit regelt d_{ST} die Robustheit der Regelung zusätzlich dazu, daß er ein Ausregelzeitknopf ist, wie unten erörtert.
- Der einzige Kriteriumsparameter, der verwendet wird, ist d_{ST}. Das elementare Kriterium lautet:

$$J_i = a_0 d_{ST} t_{ST}^s + a_1(1-d_{ST}) y_{os}^s + a_2(\sum \Delta u^2)^s$$

Hier ist t_{ST}^s die skalierte Ausregelzeit für die Simulation, y_{os}^s ist die skalierte größte anteilige Überschwingweite relative zu der Sollwertänderung und $(\Sigma\Delta u^2)^s$ ist die skalierte Summe von Regelbewegungen über den Simulationszeitraum. Natürlich variieren alle d_{ST}, t_{ST}^s, y_{os}^s und Δu mit jeder Regelkreissimulation, doch wurde der Klarheit halber die Abhängigkeit in dieser Gleichung vom Fall i nicht explizit angegeben. Die für die meisten Experimente verwendeten Gewichtungsfaktoren sind a_0: 0,75, a_1: 0,2, a_2: 0,05.

- Die Gesamtkostenfunktion J ist einfach die Summe der individuellen Werte Ji:

$$J = \sum_i J_i$$

Die skalierte Ausregelzeit t_{ST}^s basiert wie folgt auf der berechneten Ausregelzeit t_{ST}. Zunächst werden die erwarteten größten und kleinsten Werte für t_{ST} berechnet:

$$t_{ST}^{max} = A_{Tp}^{max} T_p + A_{Td}^{max} T_d$$

$$t_{ST}^{min} = A_{Tp}^{min} T_p + A_{Td}^{min} T_d$$

und dann wird t_{ST}^s innerhalb dieser Grenzen linear skaliert und begrenzt außerhalb von:

$$t_{ST}^s = \begin{cases} 1.0 & \text{falls } t_{ST} \geq t_{ST}^{max} \\ 0.0 & \text{falls } t_{ST} \leq t_{ST}^{min} \\ \dfrac{t_{ST} - t_{ST}^{min}}{t_{ST}^{max} - t_{ST}^{min}} & \text{sonst} \end{cases}$$

Ein ähnlicher Ansatz wird zum Berechnen von y_{os}^s aus y_{os} befolgt. Eine anteilige Überschwingweite von 0,0 (d.h. kein Überschwingen) wird auf einen y_{os}^s-Wert von 0,0 abgebildet, und ein anteiliges Überschwingen von 20 Prozent oder mehr (relativ zu der Sollwertänderung) wird auf 1,0 abgebildet, wobei die Skalierung dazwischen wieder linear ist. Für die quadrierte Summe von Regelbewegungen lauten diese Grenzen 1,0 und 3,0.

Die obigen Angaben haben mit einiger Ausführlichkeit spezifiziert, wie der Optimierungsalgorithmus einen Parametervektor auswertet. Dieses Verfahren ist für die Designs von Tunern für neuronale Netze für verschiedene Regler befolgt worden. Wie die Parameter für zwei Arten von Reglern - PI/PID und "lineare Regler höherer Ordnung"

(HOLC = higher order linear controllers) gewählt werden, ist unten gezeigt:

Designparameter	PID/PI-Tuner	HOLC-Tuner
r_{min}	0	0
r_{max}	0,25	0,5
A_{Tp}^{min}	0,2	0,1
A_{Tp}^{max}	2,0	2,0
A_{Td}^{min}	2,0	1,0
A_{Td}^{max}	8,0	6,0

Somit werden sowohl der PID/PI-Tuner und der HOLC-Tuner so entworfen, daß sie gestatten, daß Kompromisse zwischen Robustheit und Güte vorgenommen werden, indem einfach ein Knopf (d_{ST}) nachgestellt wird.

Es folgt ein Umriß des evolutionären Rechenalgorithmus, der für das Design der neuronalen Netztuner verwendet wird (der Parametervektor wird anstatt als p als w notiert):

1. Erzeuge eine Anfangsmenge oder Population von Vektoren $w_1, \ldots, w_N$ und bewerte Kostenfunktionswerte $J_i=J(w_i)$.

   Berechne $J_{max} = \max_i J_i$, $w_{max} = \arg\max_i J_i$

2. Wähle aus der Population willkürlich einen Parametervektor w' (gleichförmige Verteilung)
3. Wähle zufällig die Störungsstandardabweichung σ_w aus zwischen spezifizierten unteren und oberen Grenzen σ_w^{min} und σ_w^{max}
4. Erzeuge einen Störungsvektor $\delta_w \sim N(0, \sigma_w^2)$
5. Erzeuge einen Parametervektor $w'' \leftarrow w' + \delta_w$ und bewerte $J(w'')$

6. Falls $J(w'') \leq J_{max}$, dann
   Ersetze w_{max} gegen w'
   Setze w' auf w''
   Berechne erneut J_{max} und w_{max}
   Gehe zu Schritt 5 {Störung von neuem anwenden}
7. Gehe zu Schritt 2

Gemäß der vorliegenden Erfindung kann man einen Tuner entwerfen, indem man einen nichtlinearen Optimierungsalgorithmus zusammen mit einem Regelkreis-Simulationssystem verwendet. Der nichtlineare Optimierungsalgorithmus versucht, den Wert der Kostenfunktion $J(p)$ durch entsprechendes Nachstellen der Parameter p des nichtlinearen Approximators zu minimieren. Figuren 14a und 14b zeigen zwei ähnliche, auf dem nichtlinearen Approximator basierende Tuner 21. In Figur 14a ist der nichtlineare Approximator als ein neuronales Multilayer-Perzeptron-Netz implementiert. Die in diesem Fall zu optimierenden Parameter sind $w_1, \ldots, w_{14}$. In Figur 14b ist der nichtlineare Approximator als eine rechnerisch einfache Verkettungsabbildung implementiert - die Figur zeigt die mathematische Formulierung der Abbildung. Wegen seiner rechnerischen Einfachheit ist dies eine bevorzugte Ausführungsform. Die in diesem Fall zu optimierenden Parameter sind w_1, w_2, ... w_b, a_1, a_2, ..., a_b, v_{11}, v_{21}, ..., v_{b1}, v_{12}, v_{22}, ..., v_{b2}. Es sei angemerkt, daß die Implementierungen von Figuren 14a und 14b sehr ähnliche Merkmale aufweisen und in beiden Fällen K_C und K_I als Funktionen von d_{ST} und den jeweiligen Parametern ausgeben. Der in Nomura et al. beschriebene Algorithmus des überwachten Lernens kann jedoch nicht direkt für den Approximator von Figur 14b verwendet werden. Nachdem die entsprechenden Parameter bestimmt sind, um $J(p)$ zu minimieren, kann der Tuner 21 auf reale Systeme angewendet werden (vorausgesetzt, die bei dem Design angenommenen Einschränkungen gelten für das reale System).

Die Optimierung beim Tunerdesign kann kompliziert und zeitraubend erscheinen. Der Tuner 21

kann jedoch, wenn er einmal entworfen worden ist, in einer Vielfalt von unterschiedlichen Anwendungen auf einfachen Rechenplattformen leicht verwendet werden. Figur 15 zeigt, wie der Tuner 21 zur Systemregelung verwendet wird. Tuner können in verschiedenen Ausführungsformen entworfen und verwendet werden, je nach dem Charakter der Benutzerregelung der gewünschten Regelkreisleistung und den Arten von Systemen und Reglern, mit denen der Tuner verwendet wird.

Bei einem Ausführungsbeispiel eines festen linearen Prozesses erster Ordnung mit PI-Regler zum Abstimmen der Ausregelzeit kann man beschreiben, wie die vorliegende Erfindung angewendet werden kann, um einen auf einem nichtlinearen Approximator basierenden Tuner 21 für den PI-Regler 14 mit einem "Ausregelzeit"-Abstimmknopf 29 zu entwerfen, wenn von den Prozessen, für die der Tuner 21 entworfen wird, angenommen wird, daß sie durch eine lineare Dynamik erster Ordnung, die präzise bekannt ist, ausreichend charakterisiert werden. Ein System ist in Figur 15 dargestellt. In dem Laplace-Bereich wird das Prozeßmodell wie folgt ausgedrückt:

$$Y(s) = \frac{K_p}{T_p s + 1} U(s) \Rightarrow \frac{dy}{dt} = -\frac{1}{T_p} y(t) + \frac{K_p}{T_p} u(t)$$

wobei K_p die Prozeßverstärkung und T_p die Zeitkonstante ist, deren Werte bekannt sind (durch Experimente oder Prozeßwissen). Der Tuner 21 weist einen Abstimmknopf 29 d_{ST} auf, der über einen Bereich 0,0 bis 1,0 variiert werden kann, um die Prozeßantwort auf eine Sollwertänderung schnell oder langsam zu machen. Zum schnellen Ausregeln ist man hinsichtlich des Überschwingens der Antwort und hinsichtlich der Regelaktion relativ tolerant. Bei langsamem Ausregeln jedoch möchte man, daß das Überschwingen minimiert wird und die Regelaktion nicht aggressiv ist. Figur 16 zeigt eine Kurve 42 von y, wenn d_{ST} niedrig ist, und eine Kurve 43, wenn d_{ST} für eine Sollwertänderung 24 hoch

ist. Folgendes ist eine mögliche Kostenfunktion 44 (Figur 17) für diese Ausführungsform:

$$J(w) = \underset{d_{st} \in [0,1]}{E} \left((1 - d_{st}) f_{st}(t_{st}) + d_{st} f_{os}(y_{os}) + d_{st} f_{um}(\Delta u_{max}) \right) \quad (1)$$

In diesem Ausdruck bezeichnet $E(\cdot)$ den Erwartungsoperator und $d_{ST} \in [0, 1]$ bezeichnet, das die Erwartung über einen (gleichförmig verteilten) Bereich von 0 bis 1 von Werten von d_{ST} genommen werden soll. t_{ST} stellt die Ausregelzeit der Regelkreissimulation dar, y_{os} das Überschwingen in der Antwort und Δu_{max} die maximale Regelbewegung, die angetroffen wird. Die Funktionen $f_{ST}(.)$, $f_{os}(.)$ und $f_{um}(.)$ sind Skalierfunktionen, so daß der Effekt der drei Terme der relativen Wichtigkeit jedes dieser Merkmale entspricht. Man beachte, daß, da d_{ST} eine Eingabe in den Tuner 21 ist, der PI-Verstärkungen ausgibt, die das Regelkreisverhalten (und somit y_{os}, t_{st} und Δu_{max}) beeinflussen, der Wert des Ausdrucks (1) eine Funktion von Parametern p des nichtlinearen Approximators ist.

Nachdem ein Tuner 21 so entworfen ist, daß er J(w) in Gleichung (1) minimiert, kann er direkt zum Regeln des Zielprozesses verwendet werden. Die Regelgüte kann durch Verwendung des Abstimmknopfs d_{ST} 29 nachgestellt werden. Wenn d_{st} niedrig liegt (in der Nähe von 0), werden Sollwertänderungen schnell nachgeführt. Wenn d_{st} hoch ist (in der Nähe von 1), ist eine schnelle Sollwertverfolgung nicht länger wichtig, doch wird das Überschwingen der Ausgabe und die maximale Aktorbewegung minimiert. Da das Tunerdesign 21 auf einem spezifischen Prozeßmodell basiert, wird die Tunerleistung natürlich größtenteils von der Präzision des Modells abhängen. In Fällen, in denen das Modell nicht präzise bekannt ist, können Tuner für eine robuste Leistung optimiert werden.

Gleichung (1) kann in den meisten Fällen, die von praktischem Interesse sind, nicht analytisch gelöst werden. Sie kann jedoch durch Monte-Carlo-Simulationen

so präzise wie erforderlich approximiert werden. Diese Simulationsapproximation wird durch einen nichtlinearen Optimierungsalgorithmus 45 von Figur 17 geführt. Hohe Präzision wird natürlich auf Kosten der Rechenzeit erlangt.

Für die Ausführungsform eines unbestimmten linearen Prozessors erster Ordnung mit Verzögerung mit PID-Regler und Ausregelzeitabstimmung nimmt man nicht länger an, daß der Prozeß präzise bekannt ist. Der Tuner 21 wird stattdessen so entworfen, daß die Leistung über einen Bereich von Prozeßmodellen hinweg optimiert wird. Man erweitert das Prozeßmodell der Ausführungsform eines festen linearen Prozesses erster Ordnung mit PI-Regler durch Hinzufügen einer Verzögerung oder Totzeit:

$$\frac{K_p e^{-Td}}{T_p s+1}$$

wobei T_d die Prozeßtotzeit ist. Es wird angenommen, daß die drei Prozeßparameter innerhalb bekannter Räume K (für Kp), T (für Tp) und Θ (für Td) liegen. Außerdem ist man nun an einem Tuner für einen PID-Regler in interaktiver Form interessiert. Die Reglerübertragungsfunktion lautet:

$$u(s) = K_c \left(1+\frac{1}{T_i s}\right)\left(\frac{1+\tau_d s}{1+\alpha\tau_d s}\right) E(s)$$

wobei U(s) die (Laplace-transformierte) Reglerausgabe ist, T_i die integrale Zeit ist, τ_d die Differentialzeit ist, α die Geschwindigkeitsamplitude ist (eine bekannte Konstante) und E(s) der (Laplace-transformierte) Fehler ist.

In diesem Fall kann man die folgende Kostenfunktion annehmen:

$$J(w) = \max_{\substack{d_{st} \in [0,1] \\ K_p \in K \\ T_p \in T \\ T_d \in \Theta}} \left((1 - d_{st}) f_{st}(t_{st}) + d_{st} f_{ar}(y_{ar}) + d_{st} f_{um}(\Delta u_{max}) \right) \qquad (2)$$

Hier ist man, anstatt eine Erwartung zu minimieren, daran interessiert, die Worst-Case-Güte zu minimieren. Das Maximum wird über Räume für d_{st}, K_p, T_p und T_d berechnet. Die mit einem Tuner, der für die Kostenfunktion (2) optimiert ist, in Verbindung stehende Robustheit impliziert einen Verlust an nominaler Güte. Falls die Prozeßmodellparameterabschätzungen tatsächlich präziser bekannt wären, als hier angenommen wird, dann könnte eine bessere Regelung erreicht werden. Dies legt nahe, daß ein Abstimmknopf hinzugefügt werden könnte, der es dem Benutzer des Tuners gestattet, von Fall zu Fall einen Kompromiß zwischen Robustheit und nominaler Güte ohne Neuoptimierung zu treffen. Die Beschreibung der nächsten Ausführungsform erörtert diesen Gesichtspunkt der Erfindung.

Bei einer Ausführungsform eines unbestimmten linearen Prozesses erster Ordnung mit Mehrfach-Lead-Lag-Regler zur Ausregelzeit- und Robustheitsabstimmung ist der Regler nicht vom PID-Typ, sondern besteht aus drei Lead-Lag-Termen:

$$U(s) = \frac{a0\,(a1s+1)\,(a2s+1)\,(a3s+1)}{(b1s+1)\,(b2s+1)\,(b3s+1)}\,E(s)$$

Es wird ein zweiter Abstimmknopf r hinzugefügt, der auch zwischen 0 und 1 variiert. Die Parameterräume, über die der Tuner 21 hinweg arbeitet, sind nun eine Funktion von r:

$$J(w) = \underset{\substack{d_{st} \in [0,1] \\ r \in [0,1] \\ K_p \in K(r) \\ T_p \in T(r)}}{E} \left((1-d_{st}) f_{ts}(t_{st}) + d_{st} f_{os}(y_{os}) + d_{st} f_{um}(\Delta u_{max}) \right) \qquad (3)$$

Der Einfachheit halber kann man zu dem einfachen verzögerungslosen Prozeßmodell zurückkehren.

Für r=0 wird die Erwartung in diesem Fall über kleine Räume K und T bewertet, und man ist daran interessiert, die Güte auf Kosten der Robustheit zu maximieren. Wenn r erhöht wird, nehmen die Bereiche gemäß vorspezifizierter Funktionen K(r) und T(r) zu. Man beachte, daß der Effekt von r in d_{st} selbst integriert werden kann - die langsameren Ausregelzeiten, die mit hohen d_{st}-Werten verbunden sind, stimmen mit der erhöhten Robustheit überein. Somit könnte die Kostenfunktion (3) dadurch vereinfacht werden, daß r eliminiert wird und K und T von d_{st} abhängig gemacht wird.

Bei den bisher erörterten Ausführungsformen hat die Eingabe des Tuners 21 keine Informationen über den Prozeß 16 selbst enthalten. Die Tuner können in dem Ausmaß für verschiedene Prozesse verwendet werden, daß sie in die obigen Robustheitsräume K, T und Θ fallen. Es werden die nächsten beiden Ausführungsformen beschrieben, bei denen der Tuner auch Prozeßparameterabschätzungen als Eingaben erhält und er sich deshalb generisch auf einen beträchtlich größeren Raum von Prozessen anwenden läßt. Für die erste Ausführungsform wird das generische Merkmal einfach durch Skalieren von Reglerverstärkungen erhalten.

Bei einem Prozeß-generischen Tuner mit einem Prozeßmodell erster Ordnung kann man die Designbedingungen der Ausführungsform eines linearen Prozesses erster Ordnung mit einem PI-Regler annehmen und den Tuner für spezifische nominale Werte von K_p und T_p entwerfen lassen. Diese Werte sollen K_p^{nom} und

T_p^{nom} sein. Bei einer gegebenen Einstellung von d_{st} seien die von dem Tuner ausgegebenen Verstärkungen K_c^{nom} und K_i^{nom}. Die gleichen Tunerausgaben können für einen anderen Prozeß (einen, der eine Approximation erster Ordnung gestattet) mit einer Verstärkung von K_p^{new} und einer Zeitkonstanten von T_p^{new} durch Adaptieren der Verstärkungen wie folgt:

$$K_c^{new} = \frac{K_c^{nom} K_p^{nom}}{K_p^{new}}$$

$$T_i^{new} = \frac{T_i^{nom} T_p^{new}}{T_p^{nom}}$$

verwendet werden.

Skalierformeln können für alle Fälle abgeleitet werden, in denen sowohl der Regler als auch das Prozeßmodell linear sind. Ähnliche Formeln können auch für einige Fälle abgeleitet werden, in denen der Regler nichtlinear und das Prozeßmodell linear ist. Derartige Skalierformeln können willkürliche Werte von K_p und T_p kompensieren. In Fällen, in denen das Prozeßmodell zusätzliche Parameter enthält, müssen die obigen nominalen Werte Funktionen von Modellparametern sein.

Die Ausführungsform des Prozeß-generischen Tuners mit einem Prozeßmodell erster Ordnung kann man durch Hinzufügen einer Verzögerung (Totzeit) zu dem Prozeßmodell erweitern. Um einen Tuner zu entwerfen, der für alle Prozesse arbeitet, die als lineares System erster Ordnung mit Verzögerung modelliert werden können, vorausgesetzt die Verzögerung übersteigt nicht einen bestimmten konstanten Faktor β der Zeitkonstanten T_p, kann man dem Tuner eine Prozeßmodelleingabe

bereitstellen. Diese Eingabe ist $T_r=\frac{T_d}{T_p}$. Die Designkostenfunktion (1) kann nun erweitert werden als:

$$J(w)=\underset{\substack{d_{si}\in[0,1]\\ T_r\in[0,\beta]}}{E}\left((1-d_{si})f_{si}(t_{si})+d_{si}f_{os}(y_{os})+d_{si}f_{um}(\Delta u_{max})\right) \qquad (4)$$

So wird das Tunerdesign für feste Werte von K_p und von T_p ausgeführt, wobei T_d zwischen 0 und βT_p variiert. Nachdem das Design beendet ist - ein Gewichtungsvektor w ist bestimmt, der die Kostenfunktion (4) minimiert - kann der resultierende Tuner dann für jeden Prozeß verwendet werden, der als ein Prozeß erster Ordnung mit Totzeit modelliert werden kann, wobei die einzige auferlegte Bedingung lautet, daß die Totzeit nicht größer ist als das β-fache der Prozeßzeitkonstanten. Für eine derartige Verwendung müssen geschätzte Werte der Prozeßverstärkung, der Zeitkonstanten und der Totzeit in den Tuner eingegeben werden. Mit diesen wird Tr berechnet und auch Skalierungen bewirkt, wie diejenigen, die oben definiert sind. Da sich die Kostenfunktion (4) nicht mit der Robustheit beschäftigt, müssen diese Parameter präzise bekannt sein.

Bei einem Zustandsrückführungsregler mit einem nichtlinearen Prozeßmodell und Eingabestörungszurückweisungsabstimmung wird angenommen, daß der Prozeß nichtlinear ist, wobei die Modellstruktur bekannt ist:

$$\begin{pmatrix}x_1(t+1)\\ x_2(t+1)\\ x_3(t+1)\end{pmatrix}=\begin{pmatrix}2x_1(t)x_2(t)+x_1^{\,2}(t)\\ x_1(t)+x_3(t)\\ \sin(\pi x_2(t))\end{pmatrix}+\begin{pmatrix}0\\ 0\\ \gamma\end{pmatrix}u(t) \qquad (5)$$

$$y(t)=x_1(t)$$

In diesem Modell sind die x_i Zustandssystemgrößen und γ ist ein Parameter. Für eine Regelstruktur bedient man sich eines linearen Zustandsrückführungsregelgesetzes:

$$u(t)=a_1\Delta x_1(t)+a_2\Delta x_2(t)+a_3\Delta x_3(t)$$

wobei $\Delta x_i = x_i^{(d)} - x_i$, d.h. die Differenz zwischen dem gewünschten Zustandswert und dem gemessenen Wert ist. Somit besteht in diesem Fall das Abstimmproblem darin, entsprechende Werte der Reglerparameter a1, a2 und a3 zu schätzen. Dies geschieht auf der Basis von zwei Knöpfen: Einer ist eine Gütekriteriumseinstellung d_{dr}, die es dem Benutzer gestattet, bei Sprungeingabestörungen einen Kompromiß zwischen Regelaktion und schneller Störungszurückweisung zu finden. Der andere ist eine Abschätzung $\hat{\gamma}$ des Modellparameters γ. Durch Einarbeiten dieses letzteren Parameters können der gleiche Regler und Tuner über einen Bereich von Prozessen mit einer Struktur hinweg verwendet werden, die ähnlich Gleichung (5) ist, ohne auf Prozeßmodelle mit γ gleich einem bestimmten spezifischen nominalen Wert begrenzt zu sein. Die Tatsache, daß anstelle des tatsächlichen Werts ein Schätzwert von γ verwendet wird, legt nahe, daß die Abstimmungen gegenüber Fehlern bei der Parameterabschätzung eine beim Entwurf eingearbeitete Robustheit aufweisen. Folgendes ist eine geeignete Kostenfunktion für diese Ausführungsform:

$$J(w) = \underset{\substack{d_{dr} \in [0,1] \\ \gamma \in [1,5] \\ \hat{\gamma} \in [\gamma - \delta(d_{dr}), \gamma + \delta(d_{dr})] \\ d \in [-d_{max}, d_{max}] \\ x(0) \in X_0}}{E} \left((1 - d_{dr}) f_{dr}(t_{dr}) + d_{dr} f_{st}(\Sigma(\Delta u)) \right) \quad (6)$$

Werte für γ, für die dieser Tuner entworfen ist, reichen von 1 bis 5. Der Abschätzungsfehler für Abstimmeingabezwecke beträgt bis zu $\pm \delta$ und ist eine Funktion von d_{dr}, so daß für eine langsame Zurückweisungszeit eine größere Robustheit gefordert wird. Die Sprungeingabestörungen d, an deren

Zurückweisung man interessiert ist, weisen Amplituden zwischen $-d_{max}$ und d_{max} auf. t_{dr} ist die Zeit zur Zurückweisung der Eingabestörung und $\Sigma(\Delta u)$ ist die summenquadrierte Regelaktion. Die Erwartung muß ebenfalls über einen Raum x_0 von Anfangsbedingungen x(0) berechnet werden.

Für die Ausführungsform eines Zustandsrückführungsreglers mit einem nichtlinearen Prozeßmodell und Eingabestörungsabstimmung wie oben kann man auch einen nichtlinearen Regler bekannter Struktur verwenden, beispielsweise:

$$u(t) = a_1x_1(t) + a_2x_2(t) + a_3x_3(t) + a_4x_1(t)x_2(t) + a_5x_1^2(t)\,x_3(t) + a_6x_2(t)x_3(t)$$

Bei dieser Ausführungsform gibt der Tuner sechs Parameter a_1-a_6 aus. Die Kostenfunktion (6) kann wieder verwendet werden.

Für einen PD-Regler mit integrierendem linearem Prozeß, leicht zu berechnenden Antwortmerkmalseingaben und Anregelzeitabstimmung kann folgendes implementiert werden.

Das Prozeßmodell lautet:

$$Y(s) = \frac{K_p}{s(T_ps+1)}\,U(s)$$

Der Regler ist:

$$u(t) = K_ce(t) + K_d\frac{e(t) - e(t-\Delta t)}{\Delta t}$$

Tunerausgaben sind Proportionalverstärkung K_c und Differentialverstärkung K_d. Die Tunereingabe besteht aus einfachen, aus der Prozeßausgabe berechneten Merkmalen nach dem Austritt aus dem Prozeß zum Zeitpunkt t_0 mit einem Impuls der Höhe H und Breite

W. Die Prozeßausgabe wird zu den Zeitpunkten $t_0+n\frac{W}{4}$, für n=0, 1, ..., 10 gemessen. Zehn Merkmale werden einfach als $y(t_0+n\frac{W}{4})- y(t_0)$ für n=1, 2, ..., 10 berechnet. Bei dieser Ausführungsform wird weder von Tp noch von Kp angenommen, daß sie bekannt sind. Entsprechende Werte für H und W hängen jedoch tatsächlich von dem Prozeß ab, und einige allgemeine Begriffe der Zeitkonstanten und der Verstärkung sind erforderlich.

In allen obigen Ausführungsformen ist das Schlüsselelement des Tuners ein nichtlinearer Approximator. Verschiedene nichtlineare Approximatoren können in diesem Kontext verwendet werden, einschließlich neuronale Multilayer-Perzeptron-Netze, rechnerisch einfache sigmoidale Verkettungen, Radial-Basisfunktionsnetze, Funktionsverknüpfungsnetze, CMAC-Netze, Fuzzy-Logik-Modelle, die Fuzzifizierung und/oder Defuzzifizierung und/oder Zugehörigkeitsfunktionen verwenden, Wavelet-Netze, Polynomentwicklungen und spezifische nichtlineare parametrisierte Strukturen.

ANSPRÜCHE

1. Autotuner (21) zum Abstimmen eines Reglers (14), der folgendes umfaßt:

Ein Vorverarbeitungsmittel mit einer Eingabe und einer Ausgabe zum Umwandeln einer Menge von Eingabeparametern am Eingang in eine Menge von normierten Parametern am Ausgang,

ein nichtlineares Approximatormittel (28) mit einem mit dem Ausgang des Vorverarbeitungsmittels verbundenen Eingang zum algorithmischen Operieren an der Menge normierter Parameter,

wobei das nichtlineare Approximatormittel offline ohne vorheriges überwachtes Lernen eingerichtet wird, wobei das Einrichten ein System verwendet mit mehreren Regelkreissimulationen und einem Optimierungsalgorithmus (45),

wodurch während des Einrichtens der Optimierungsalgorithmus den Wert einer Kostenfunktion (44) minimiert, die von den mehreren Regelkreissimulationen abgeleitet ist und die die Güte der Simulationen angibt, um optimierte Parameter des nichtlinearen Approximatormittels zu erzeugen,

wobei die Ausgabe des nichtlinearen Approximators eine Menge von auf der Grundlage des minimierten Werts der Kostenfunktion abgeleiteten normierten Abstimmungsparametern ist; und

ein Nachverarbeitungsmittel mit einem ersten, mit dem Ausgang des nichtlinearen Approximatormittels verbundenen ersten Eingang, mit einem mit dem Eingang des Vorverarbeitungsmittels verbundenen zweiten Eingang und mit einem Ausgang zum Skalieren der Menge normierter Abstimmungsparameter in Reglerabstimmungsparameter (22) am Ausgang.

2. Tuner nach Anspruch 1, wobei:

die Eingabeparameter entweder von dem einen oder von beiden der zwei Arten von Parametern sind,

wobei die beiden Arten von Parametern sind:

Parameter (26), die ein System als geregelt charakterisieren; und

Parameter (27), die ein gewünschtes Regelkreisverhalten charakterisieren, und

das nichtlineare Approximatormittel offline ohne überwachtes Lernen optimiert wird, und wobei die Offline-Optimierung keine Erzeugung von optimierten Abstimmungsparametern zum Optimieren des nichtlinearen Approximatormittels erfordert.

3. Tuner nach Anspruch 2, wobei das nichtlineare Approximatormittel ein neuronales Netz ist.

4. Tuner nach Anspruch 2, wobei das nichtlineare Approximatormittel ein Fuzzy-Approximator ist.

5. Tuner nach Anspruch 2, wobei das nichtlineare Approximatormittel ein Radial-Basisfunktionsapproximator ist.

6. Tuner nach Anspruch 2, wobei das nichtlineare Approximatormittel ein Approximator ist, der nichttranszendente mathematische Funktionen verwendet.

7. Tuner nach Anspruch 2, wobei das nichtlineare Approximatormittel ein Approximator ist, der rechnerisch einfache sigmoidale Verkettungsabbildungen verwendet.

8. Autotuner (21) zum Abstimmen eines Reglers (14), der folgendes umfaßt:

einen nichtlinearen Approximator (28), dessen Verhalten durch Nachstellen seiner Parameter modifiziert wird und der offline ohne überwachtes vorheriges Lernen eingerichtet wird, wobei das Einrichten ein System verwendet, das einen Regelkreissimulator und einen Optimierungsalgorithmus (45) verwendet,

wobei während des Einrichtens der Regelkreissimulator Reglerparameter von dem Regler zur Verwendung in mehreren Regelkreissimulationen empfängt, wobei die Regelkreissimulationen die Berechnung des Werts einer Kostenfunktion (44) ermöglichen, die die Güte der Regelkreissimulationen angibt,

wodurch der Wert der Kostenfunktion durch den Optimierungsalgorithmus verarbeitet wird, der optimierte nichtlineare Approximatorparameter erzeugt, die auf derartige Weise optimiert werden, daß der Wert der Kostenfunktion minimiert wird,

wobei der minimierte Wert der Kostenfunktion zum Ableiten von entsprechenden Reglerabstimmungsparametern (22) zur Lieferung an den Regler verwendet wird, die zu einer optimierten Reglergüte führen.

9. Tuner nach Anspruch 8, wobei das nichtlineare Approximatormittel ein neuronales Netz ist.

10. Tuner nach Anspruch 8, wobei das nichtlineare Approximatormittel ein Fuzzy-Approximator ist.

11. Tuner nach Anspruch 8, wobei das nichtlineare Approximatormittel ein Radial-Basisfunktions-approximator ist.

12. Tuner nach Anspruch 8, wobei das nichtlineare Approximatormittel ein Approximator ist, der nichttranszendente mathematische Funktionen verwendet.

13. Tuner nach Anspruch 8, wobei das nichtlineare Approximatormittel ein Approximator ist, der rechnerisch einfache sigmoidale Verkettungsabbildungen verwendet.

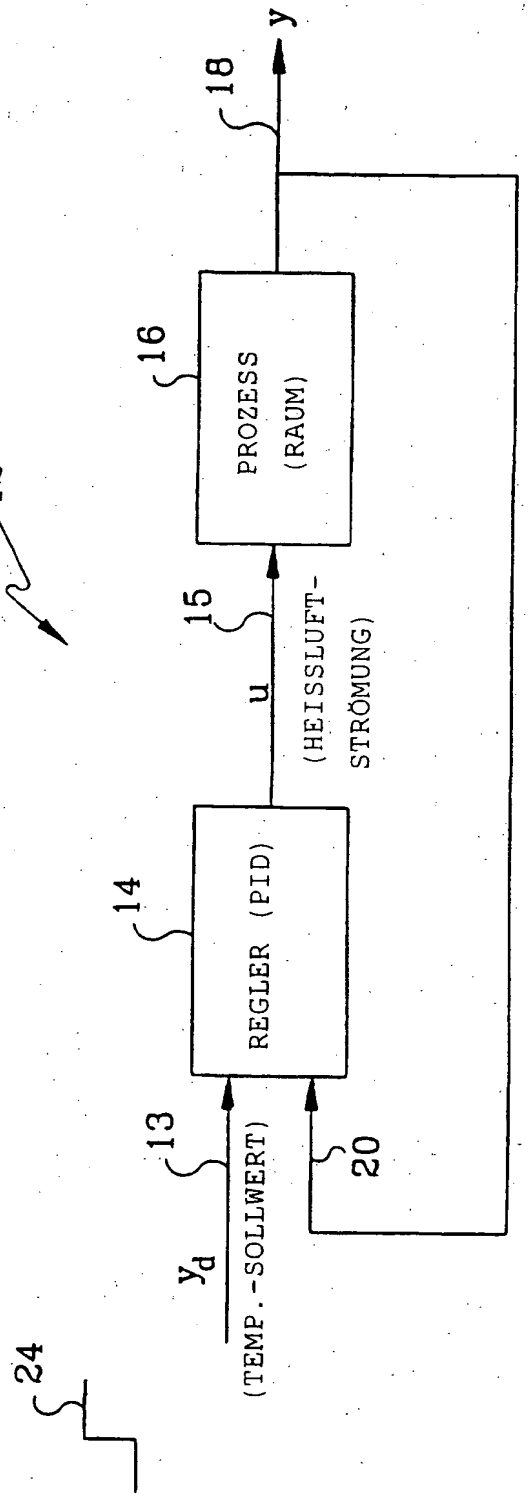

12
24
13
y_d
(TEMP.-SOLLWERT)
14
REGLER (PID)
15
u
(HEISSLUFT-STRÖMUNG)
16
PROZESS (RAUM)
18
y
20

Fig. 1

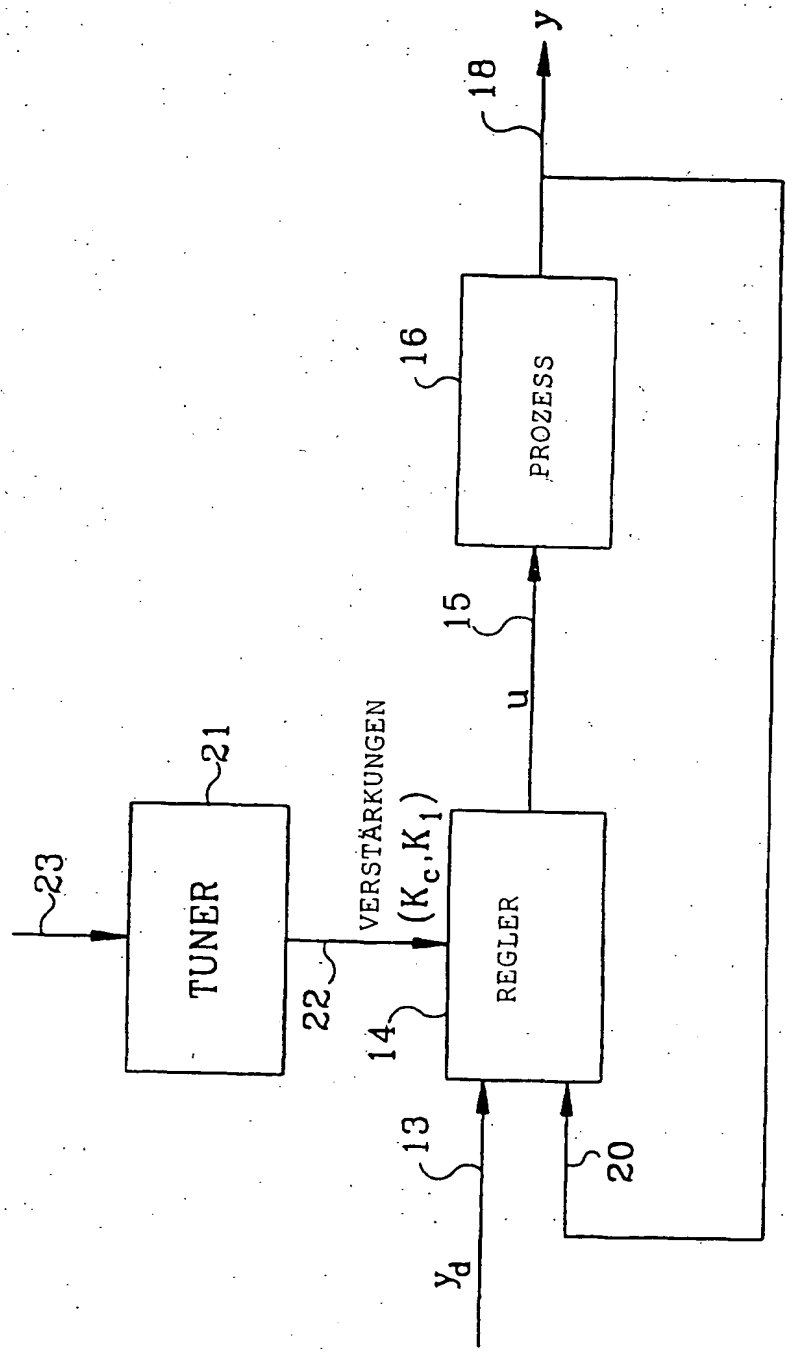

y
18
PROZESS
16
15
u
VERSTÄRKUNGEN
(K_c, K_I)
REGLER
14
22
TUNER
21
23
13
y_d
20

Fig.2

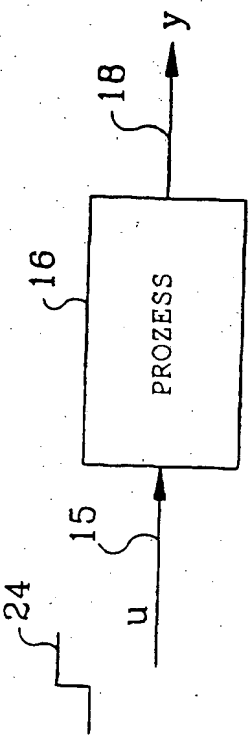


Fig.3

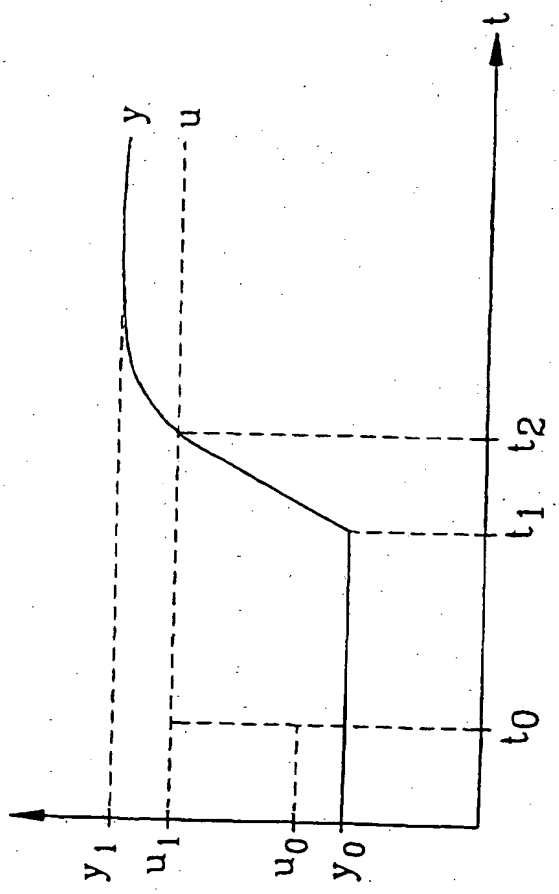


Fig.4

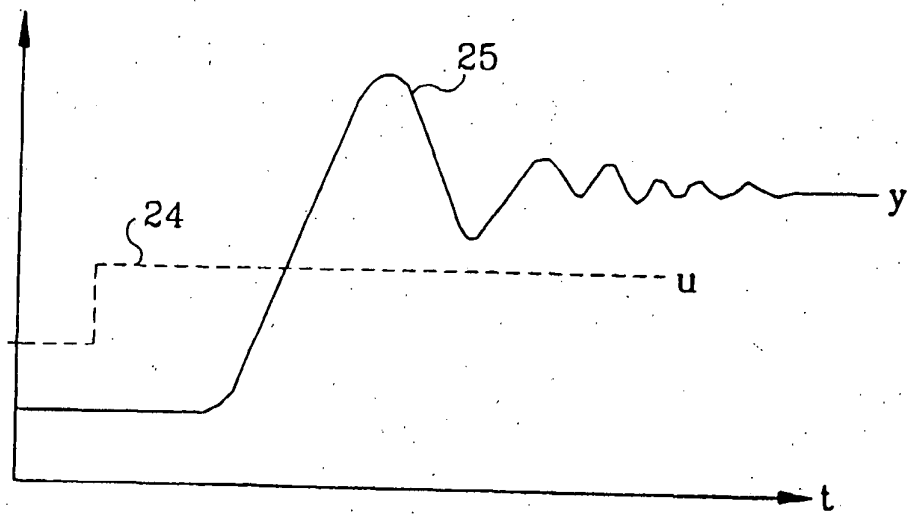


Fig.5

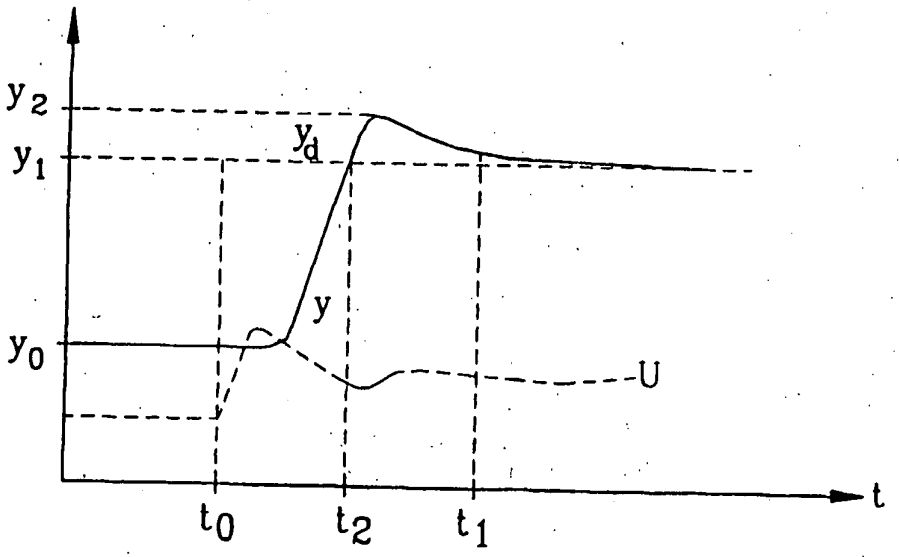


Fig.6

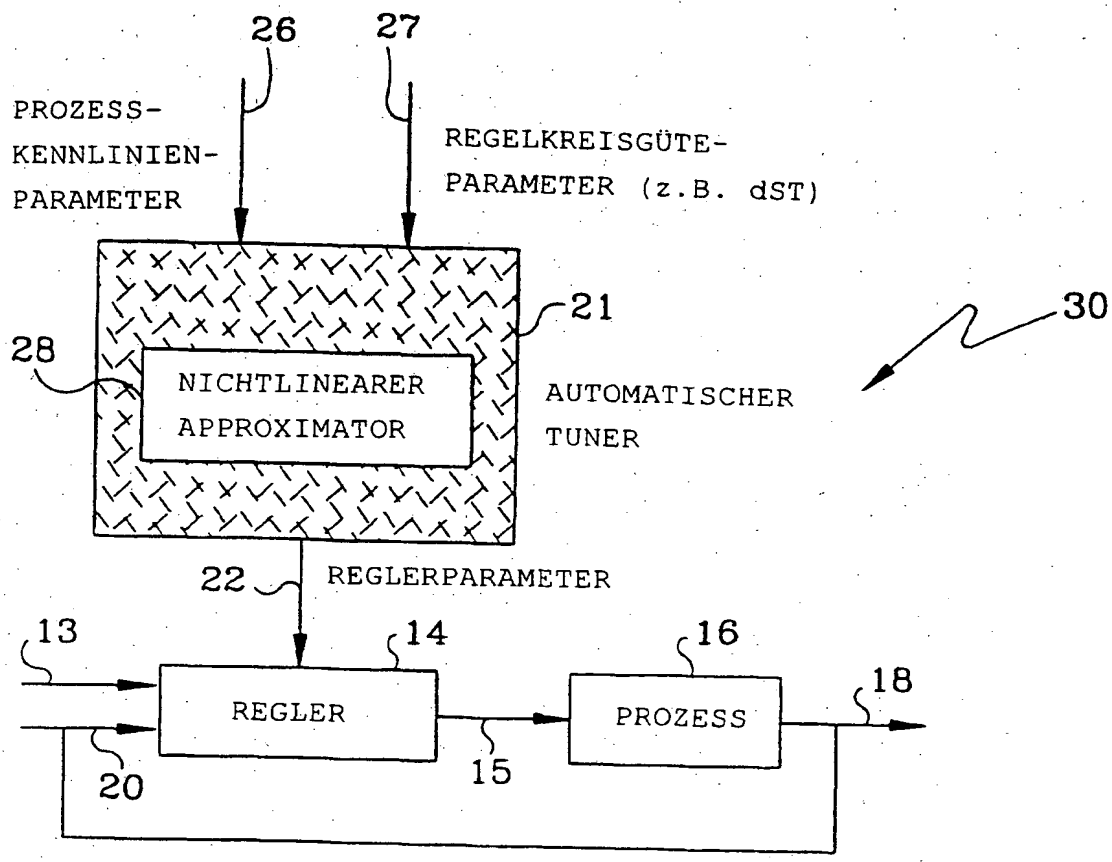

26
27
PROZESS-
KENNLINIEN-
PARAMETER
REGELKREISGÜTE-
PARAMETER (z.B. dST)
21
30
28
NICHTLINEARER
APPROXIMATOR
AUTOMATISCHER
TUNER
22
REGLERPARAMETER
13
14
16
18
REGLER
PROZESS
20
15

Fig.7

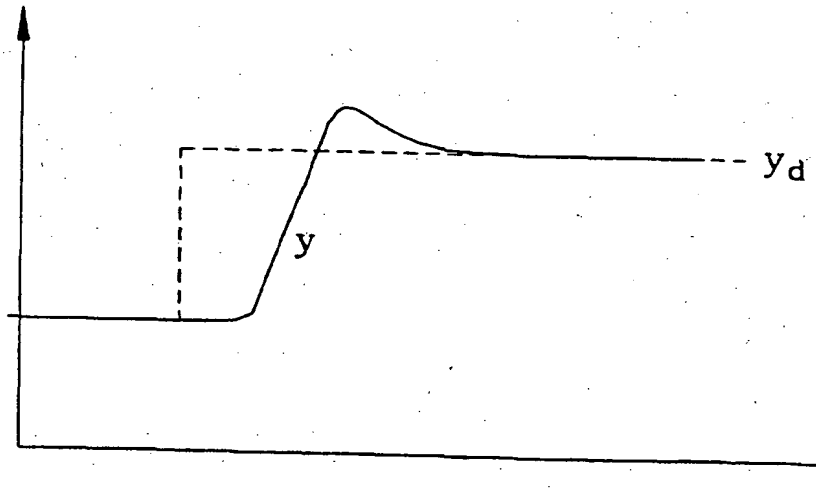

y_d
y
t

Fig.8

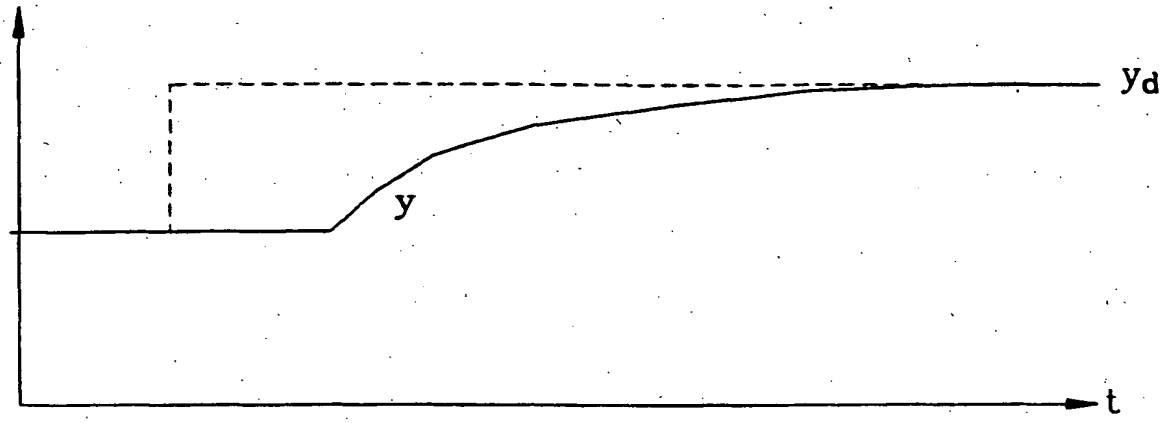


Fig. 9

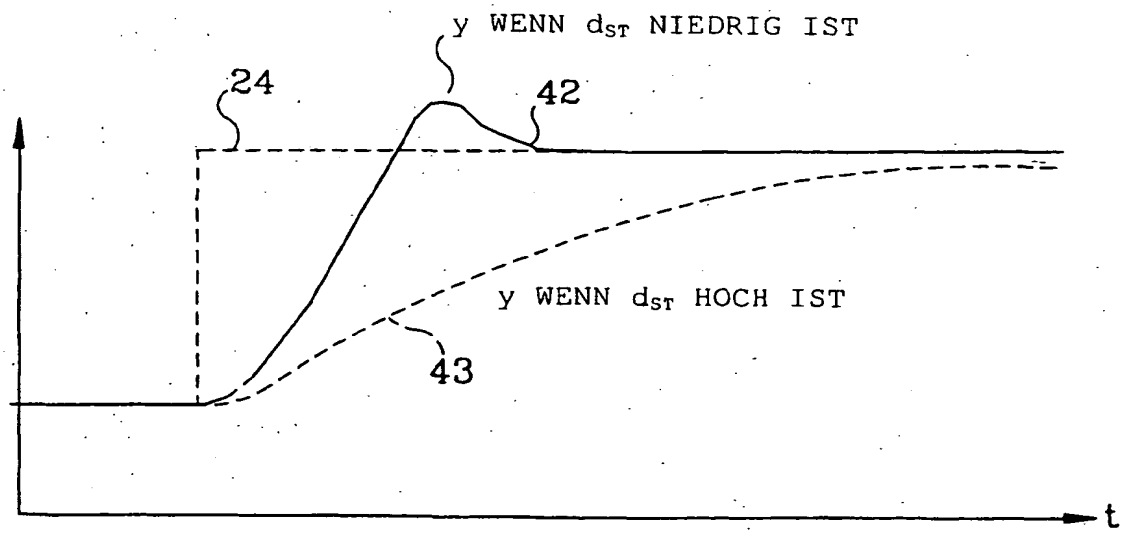


Fig. 16

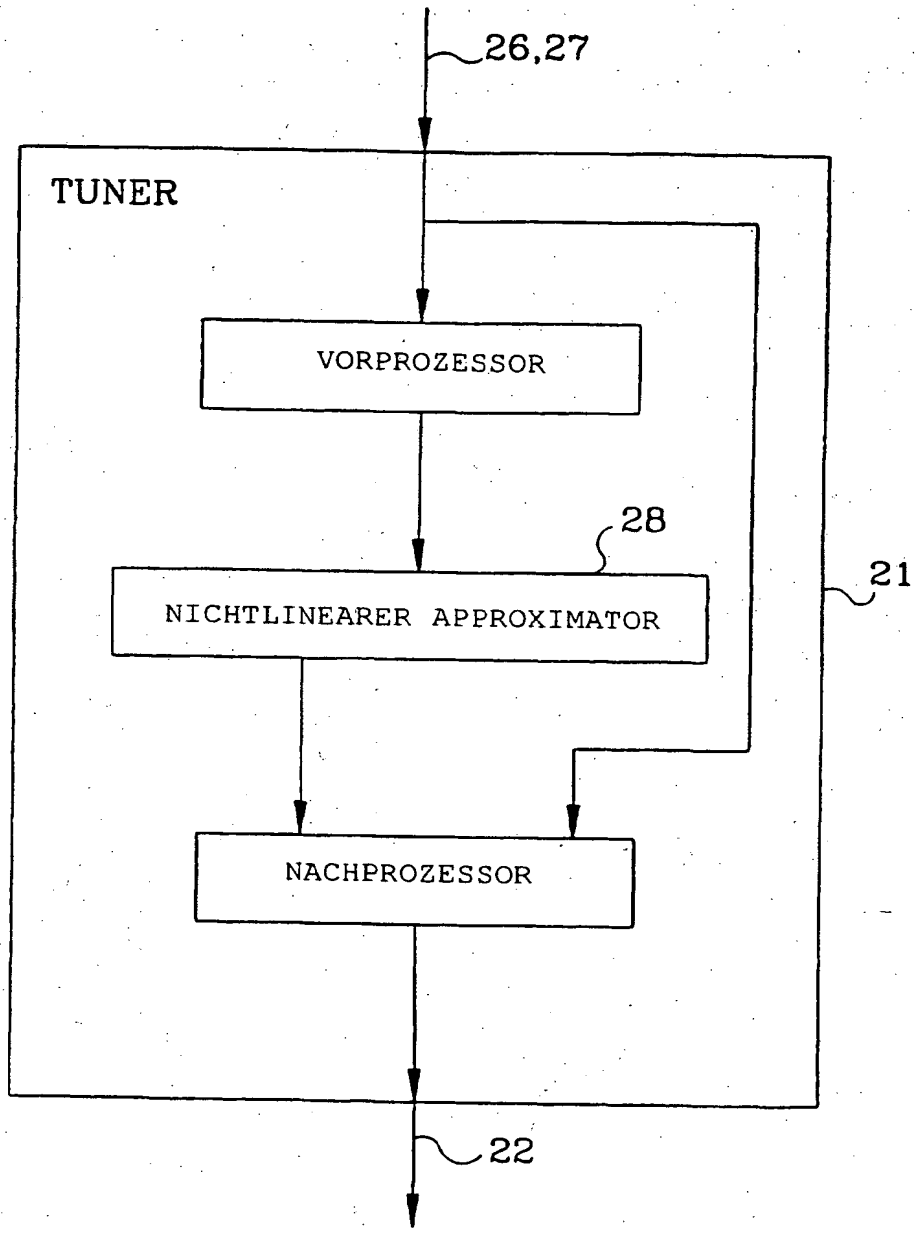

26,27
TUNER
VORPROZESSOR
28
NICHTLINEARER APPROXIMATOR
21
NACHPROZESSOR
22

Fig. 10a

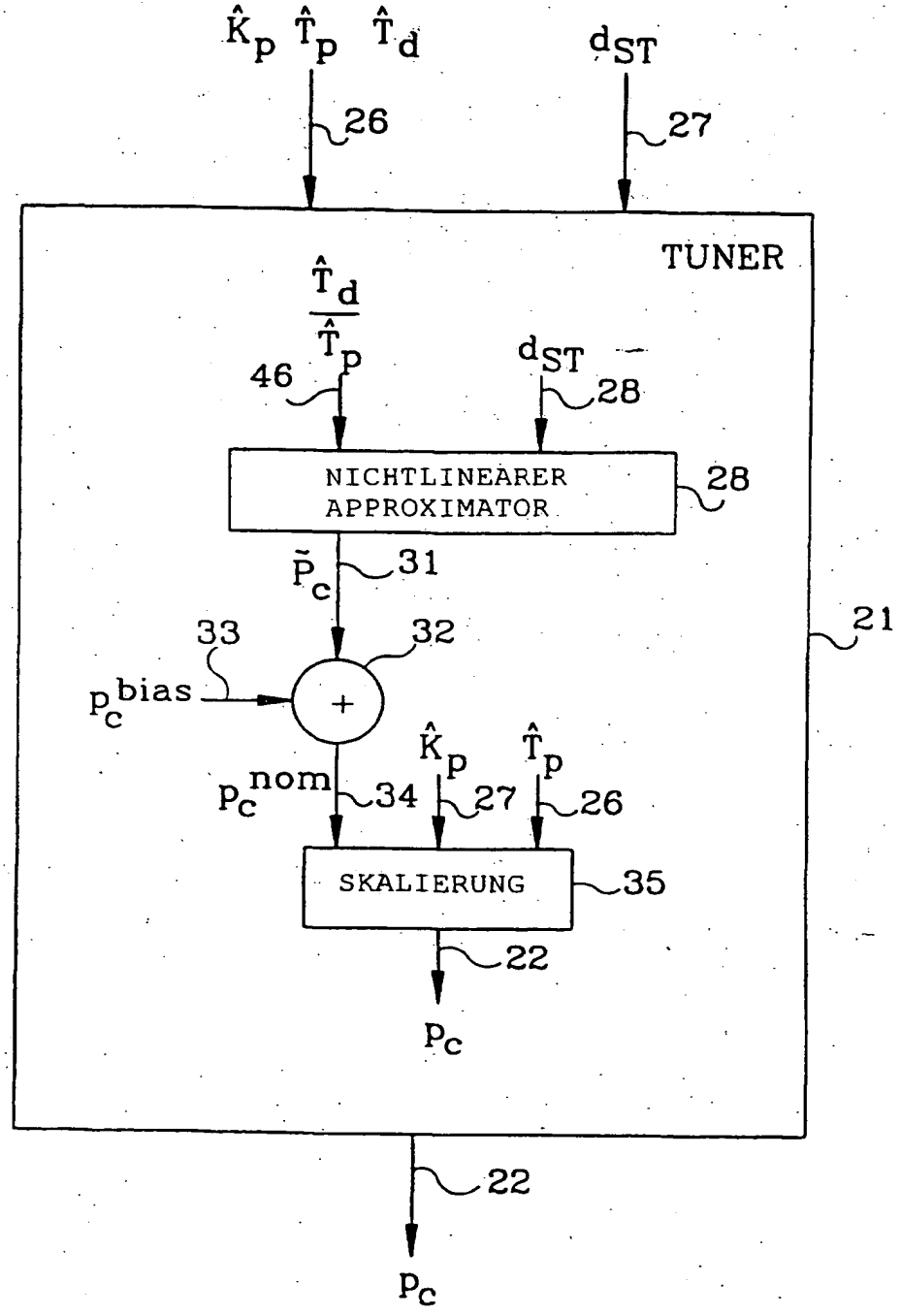

$\hat{K}_p$ $\hat{T}_p$ $\hat{T}_d$
26
d_{ST}
27
TUNER
$\frac{\hat{T}_d}{\hat{T}_p}$
46
d_{ST}
28
NICHTLINEARER APPROXIMATOR
28
$\tilde{P}_c$
31
21
33
32
p_c^{bias}
+
$\hat{K}_p$
$\hat{T}_p$
p_c^{nom}
34
27
26
SKALIERUNG
35
22
p_c
22
p_c

Fig. 10b

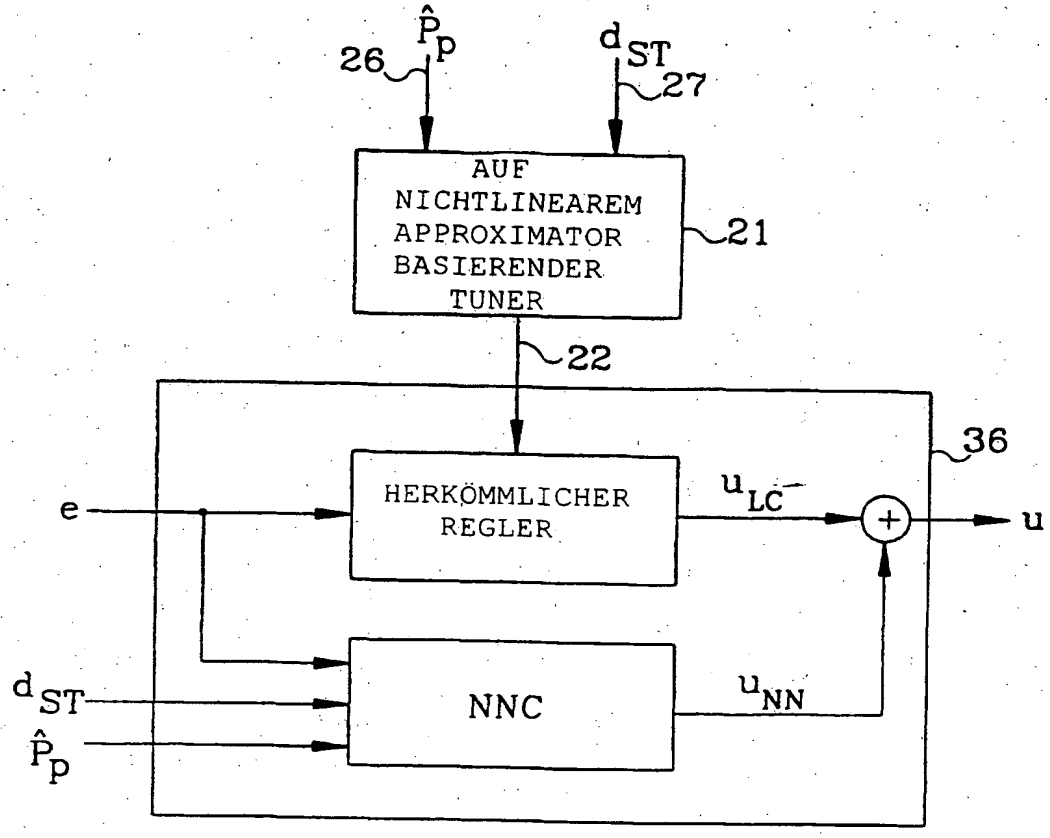

$\hat{P}_p$
26
d_{ST}
27
AUF NICHTLINEAREM APPROXIMATOR BASIERENDER TUNER
21
22
36
e
HERKÖMMLICHER REGLER
u_{LC}
+
u
NNC
u_{NN}
d_{ST}
$\hat{P}_p$

Fig. 11

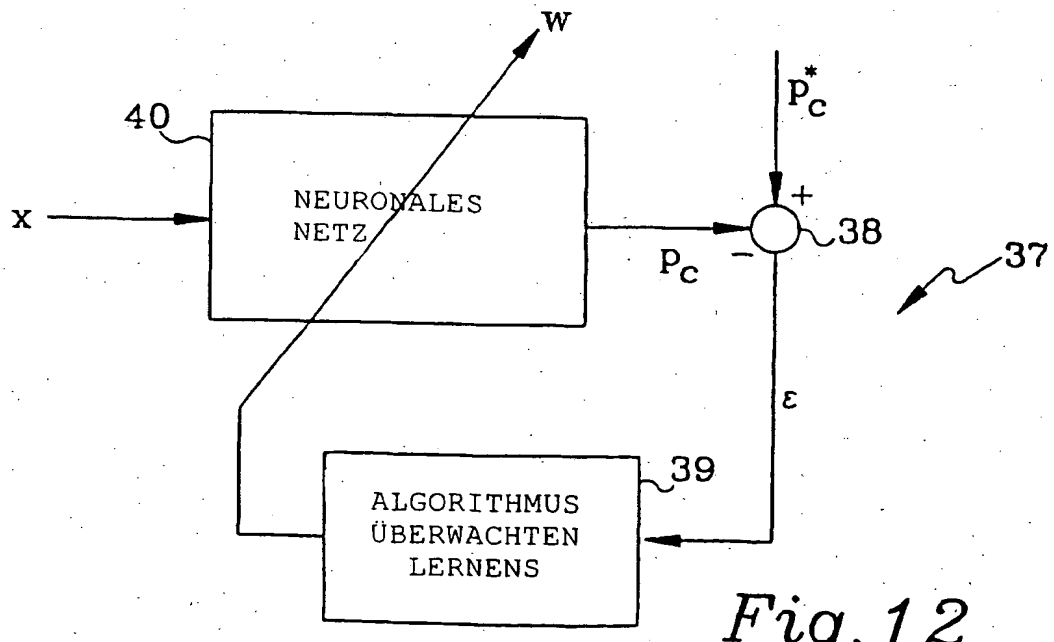

w
40
P_c^*
x
NEURONALES NETZ
+
38
P_c
−
37
ε
39
ALGORITHMUS ÜBERWACHTEN LERNENS

Fig. 12

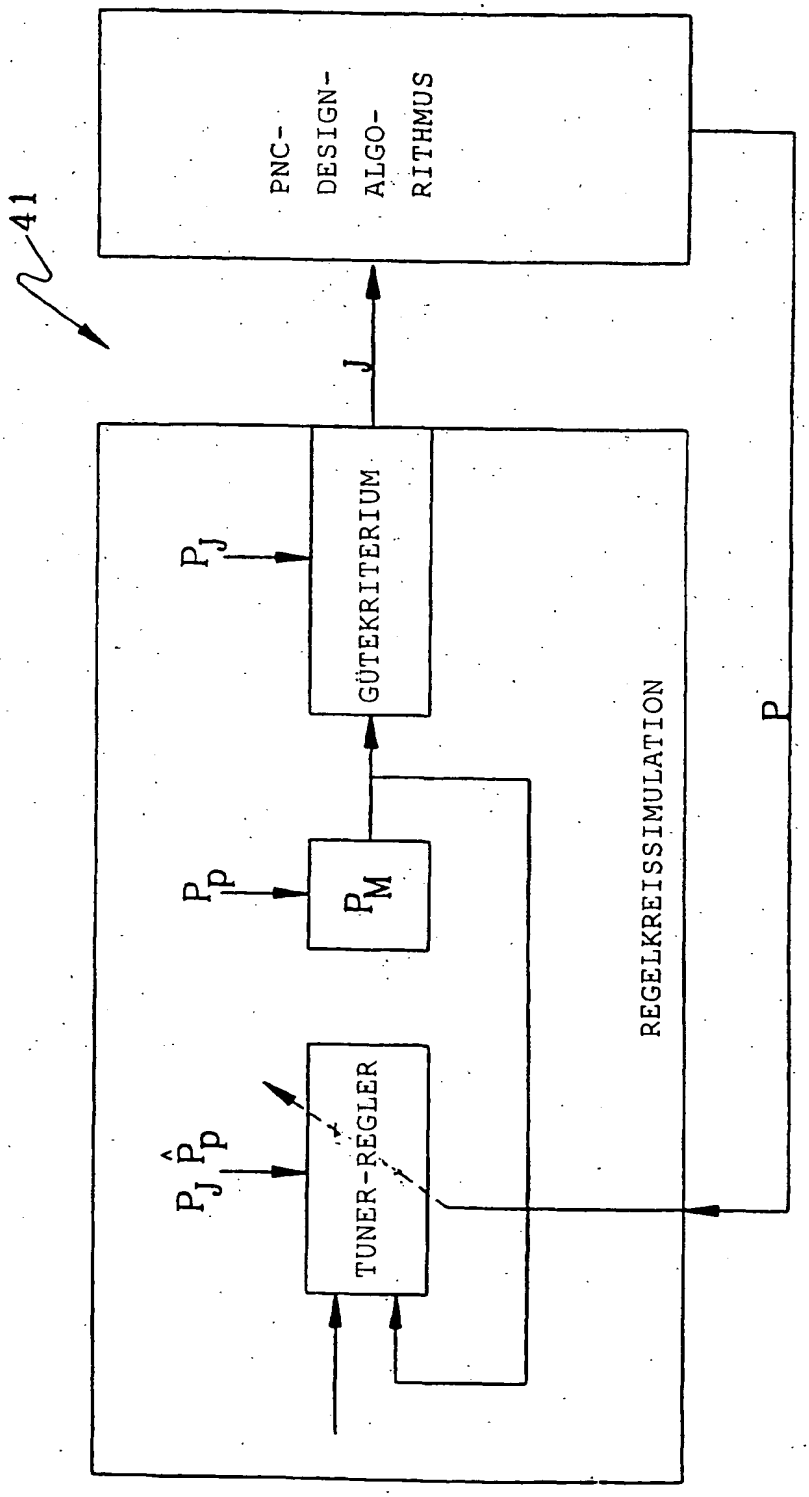

41
PNC-
DESIGN-
ALGO-
RITHMUS
J
GÜTEKRITERIUM
P_J
P_M
P_p
TUNER-REGLER
P_J $\hat{P}_p$
REGELKREISSIMULATION
P

Fig. 13

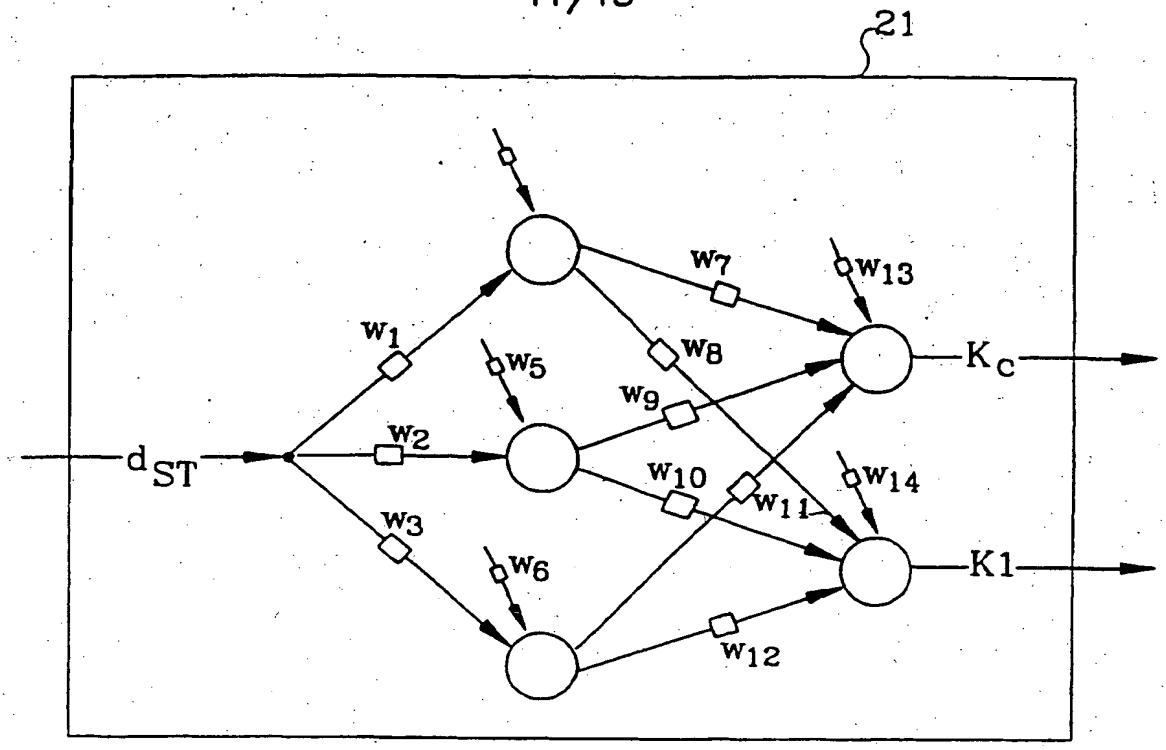


Fig. 14a

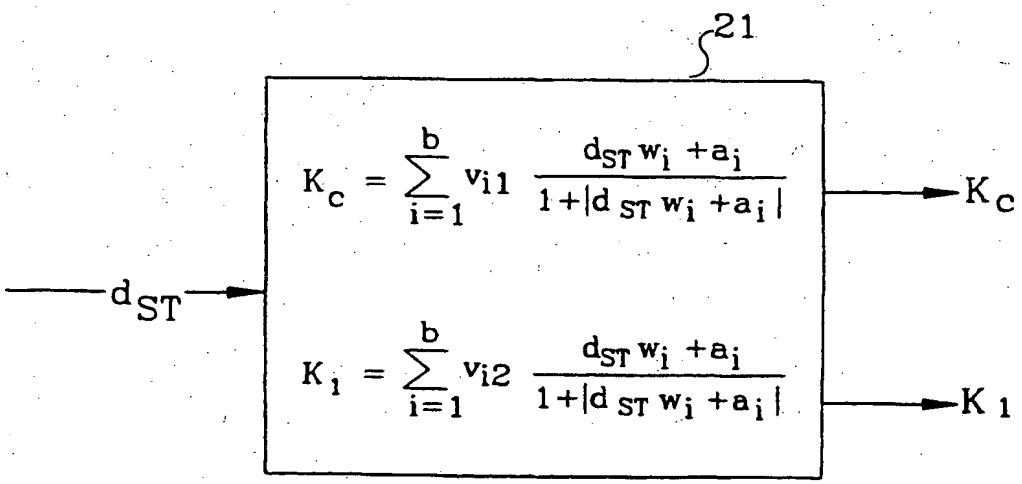


Fig. 14b

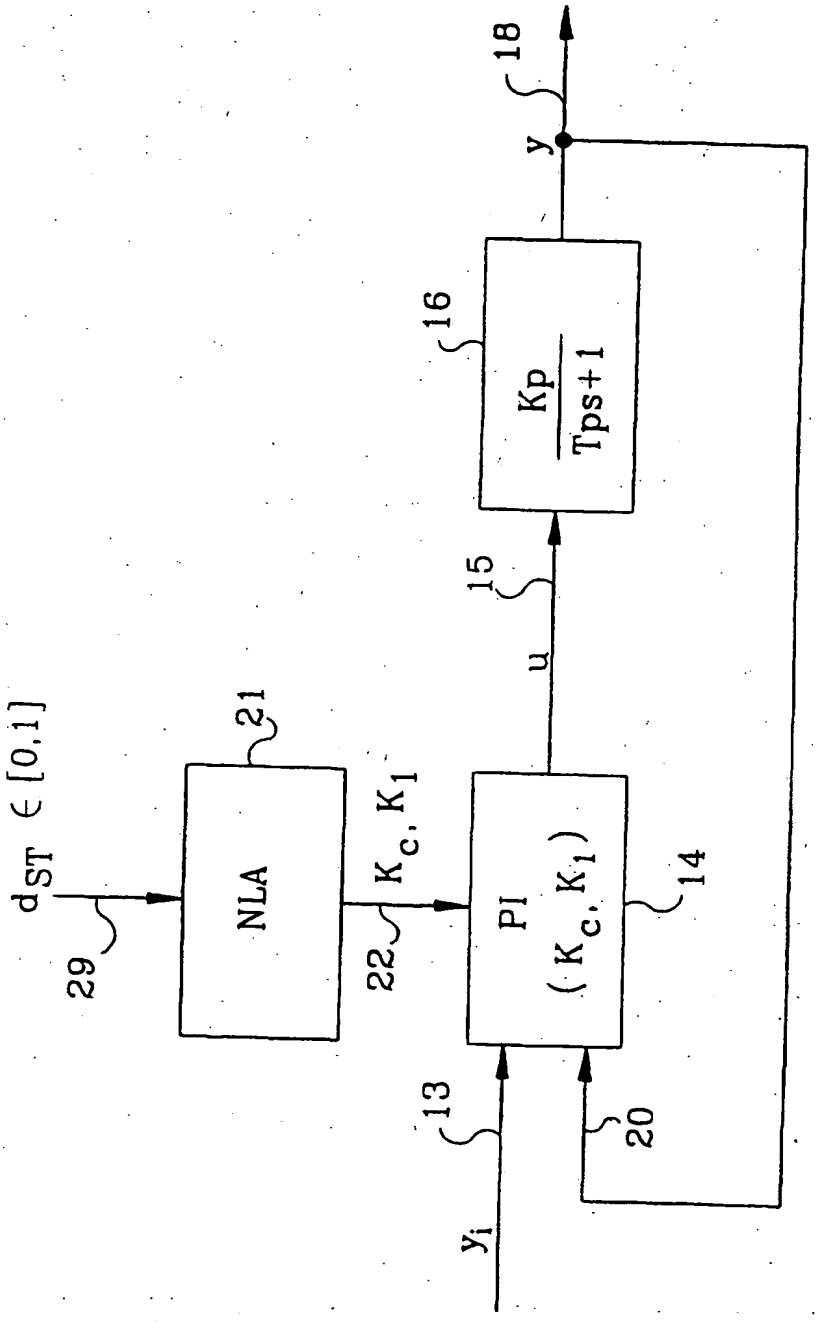
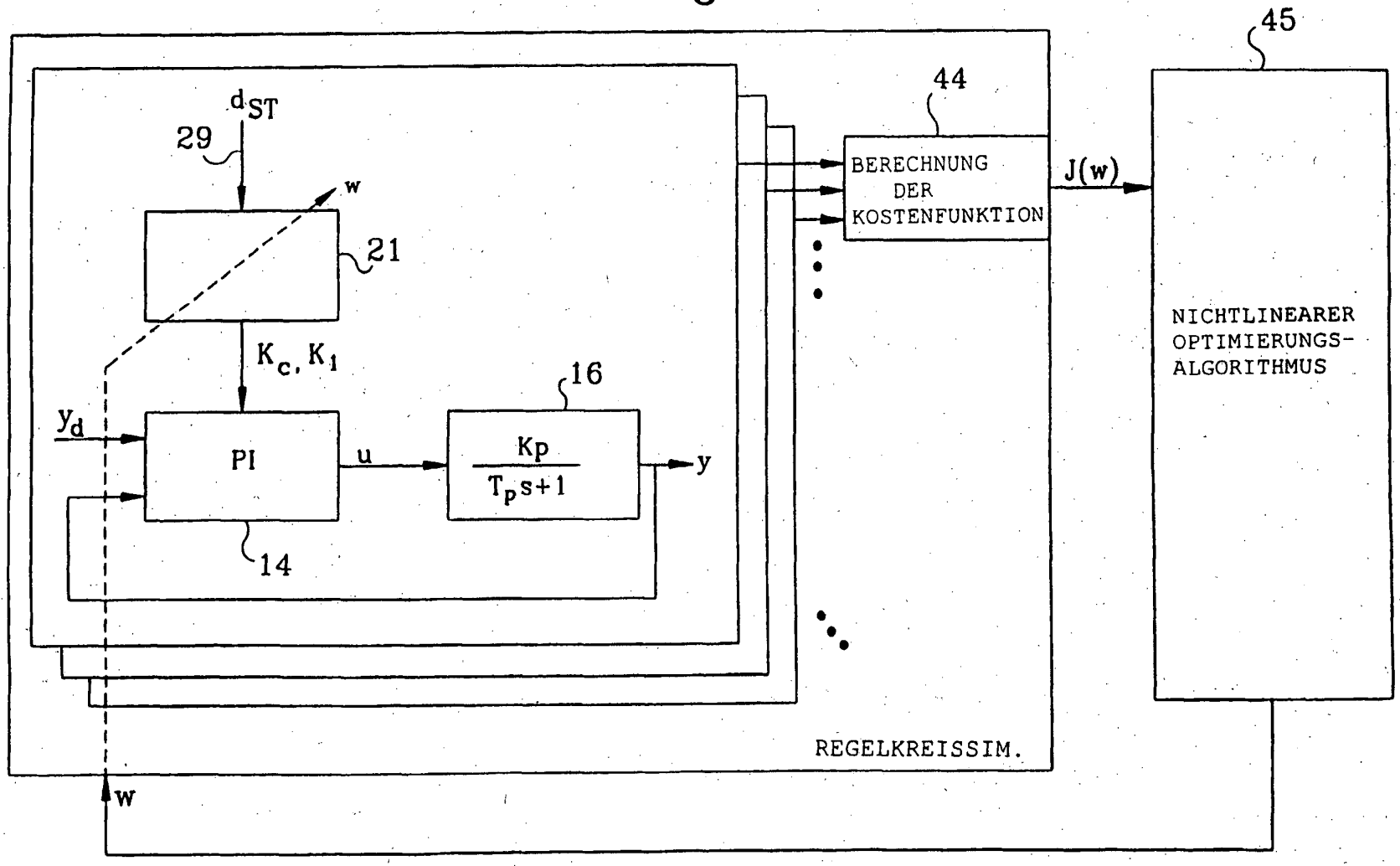

$d_{ST} \in [0,1]$
29
NLA
21
K_c, K_1
22
PI
(K_c, K_1)
14
u
15
$\frac{Kp}{Tps+1}$
16
y
18
y_i
13
20

Fig. 15

Fig. 17


45
44
d_{ST}
29
w
21
K_c, K_1
16
y_d
PI
u
Kp
$T_p s+1$
y
14
BERECHNUNG DER KOSTENFUNKTION
J(w)
NICHTLINEARER OPTIMIERUNGS-ALGORITHMUS
REGELKREISSIM.
w